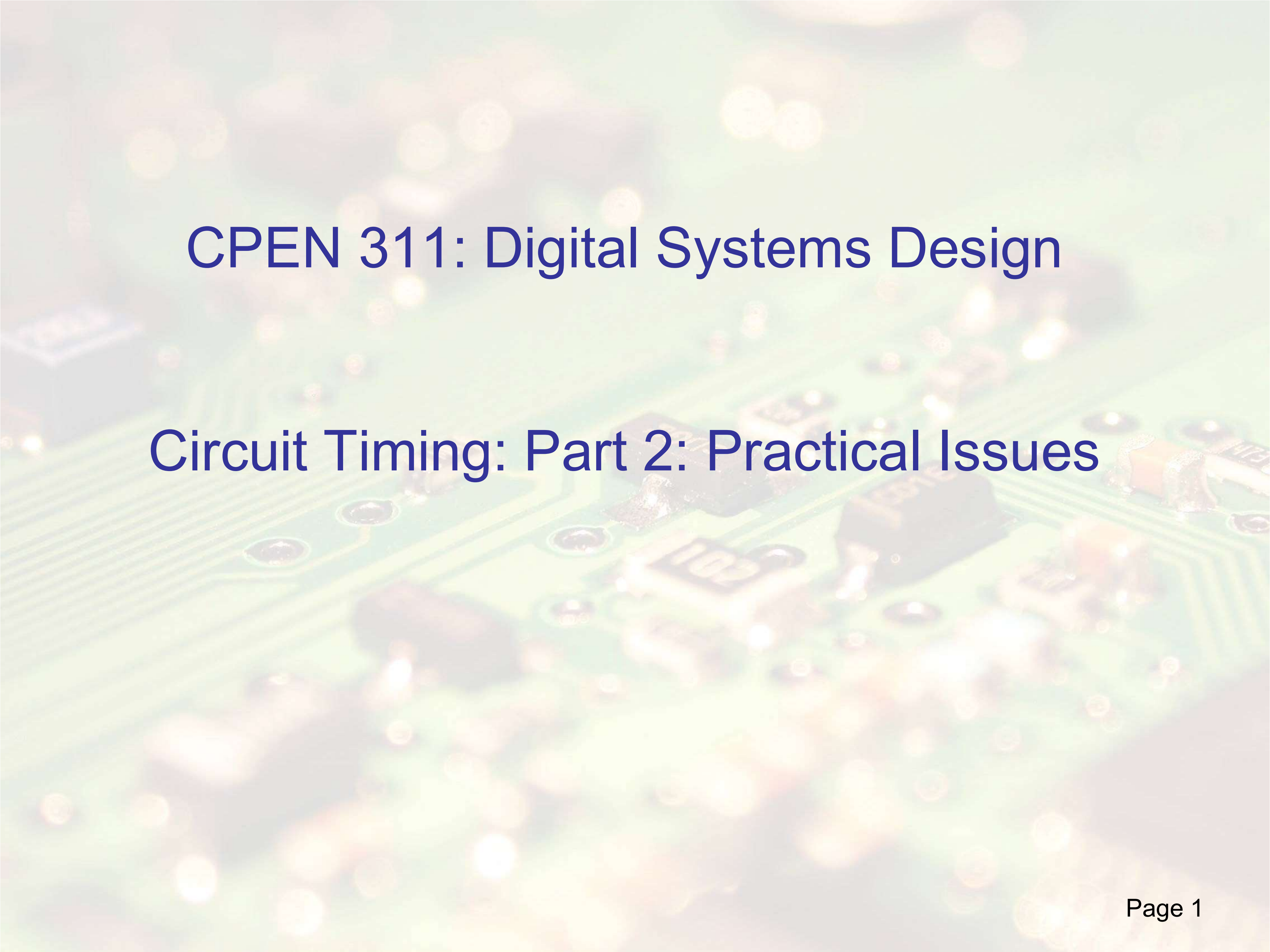


CPEN 311: Digital Systems Design

Circuit Timing: Part 2: Practical Issues

Learning Objectives

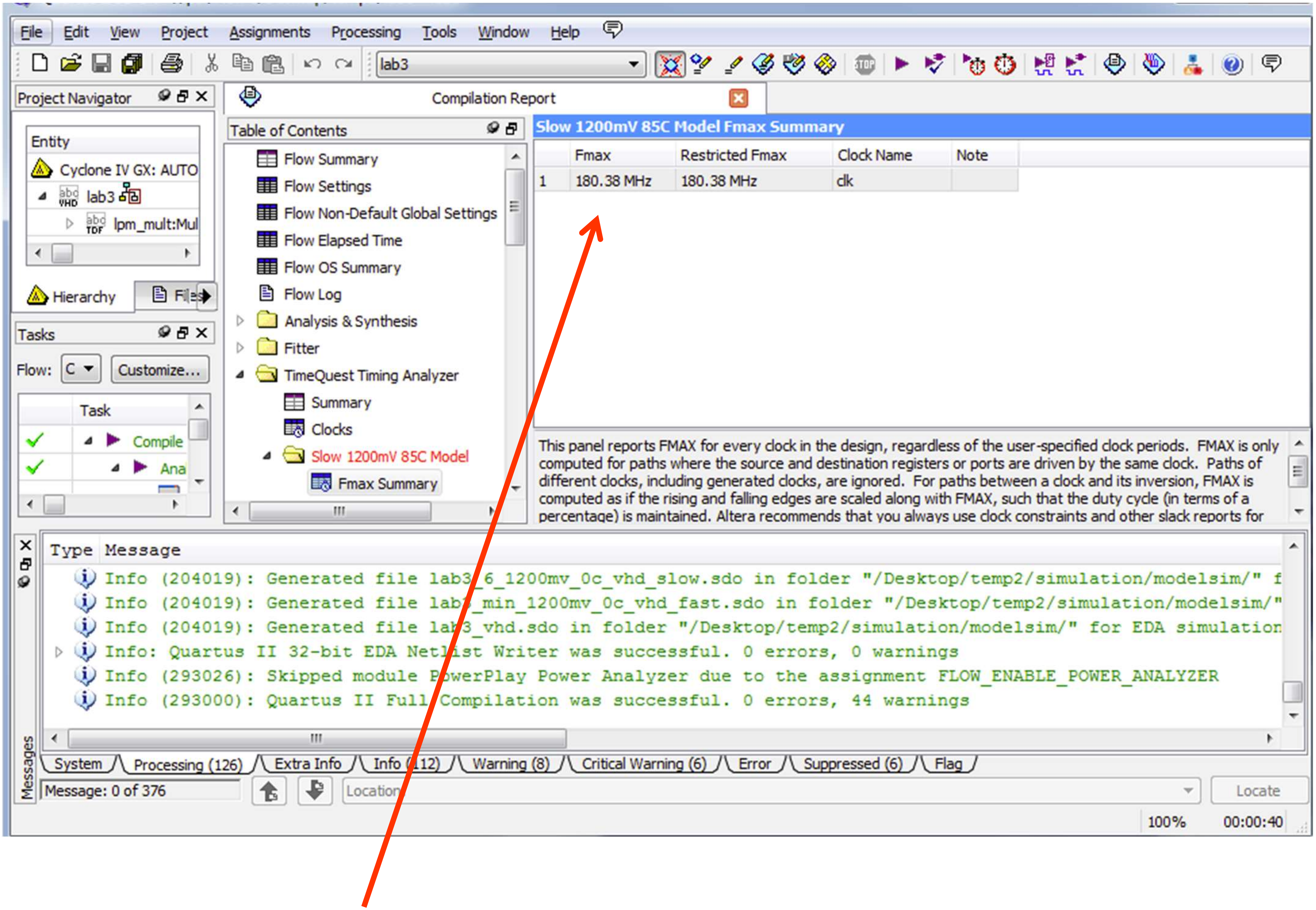
1. Understand what timing closure is and why it is difficult
2. Understand how pipelining can help with timing closure
3. Understand retiming and be able to apply it to a circuit
4. Be able to discuss the effects of clock skew
5. Understand what a PLL is used for
6. Understand the cause and impact of glitches caused by unequal combinational path delays

Timing Closure

Timing Closure: Ensuring your design meets timing constraints

So far you have been compiling and just hoping it runs at the required speed
(eg. If you are using CLOCK_50, the critical path <= 20 ns)

That was fine for these simple labs, but for complex designs, a simple compile may not lead to you meet your timing constraints.



Slow 1200mV 85C Model Fmax Summary

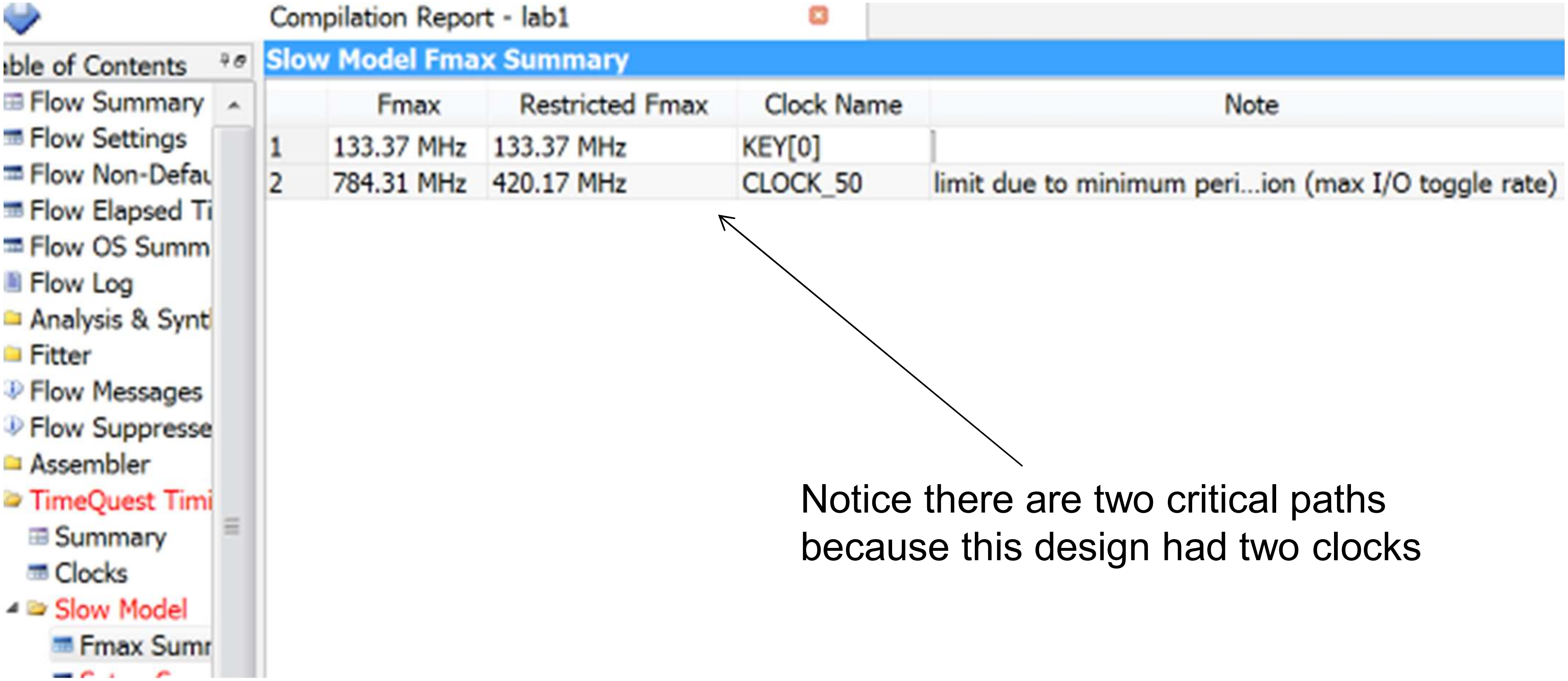
	Fmax	Restricted Fmax	Clock Name	Note
1	180.38 MHz	180.38 MHz	clk	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if the rising and falling edges are scaled along with FMAX, such that the duty cycle (in terms of a percentage) is maintained. Altera recommends that you always use clock constraints and other slack reports for

```
Type Message
     Info (204019): Generated file lab3_6_1200mv_0c_vhd_slow.sdo in folder "/Desktop/temp2/simulation/modelsim/" f
     Info (204019): Generated file lab3_min_1200mv_0c_vhd_fast.sdo in folder "/Desktop/temp2/simulation/modelsim/"
     Info (204019): Generated file lab3_vhd.sdo in folder "/Desktop/temp2/simulation/modelsim/" for EDA simulation
     Info: Quartus II 32-bit EDA Netlist Writer was successful. 0 errors, 0 warnings
     Info (293026): Skipped module PowerPlay Power Analyzer due to the assignment FLOW_ENABLE_POWER_ANALYZER
     Info (293000): Quartus II Full Compilation was successful. 0 errors, 44 warnings
```

Based on register-register delays

From my implementation of the lab:



Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	133.37 MHz	133.37 MHz	KEY[0]	
2	784.31 MHz	420.17 MHz	CLOCK_50	limit due to minimum peri...ion (max I/O toggle rate)

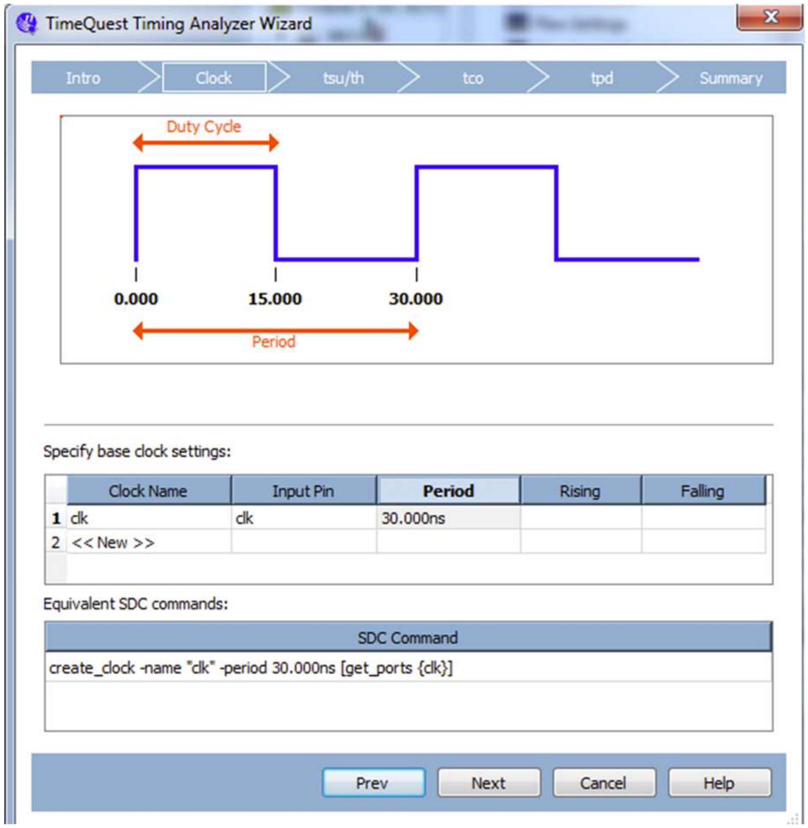
Notice there are two critical paths because this design had two clocks

Fmax means the maximum frequency the circuit can run at

Timing Constraints in Quartus II

You can specify a desired target clock frequency aka **Timing Constraint**

Assignments Menu → Timing Analysis Wizard



Specify base clock settings:

	Clock Name	Input Pin	Period	Rising	Falling
1	clk	clk	30.000ns		
2	<< New >>				

Equivalent SDC commands:

SDC Command
create_clock -name "clk" -period 30.000ns [get_ports {clk}]

Quartus Timing Optimization

- Tries to meet your desired clock frequency
  - By default, an unachievable value (eg, 1GHz)
  - Better to specify actual target (eg, 50MHz)
- May not meet target
  - Limited by logic + routing delays in FPGA

When you ask for…

slow clock frequency

smaller circuits → optimize total number of logic gates

fast clock frequency

larger circuit → optimize number of logic gates along path

All about tradeoffs!

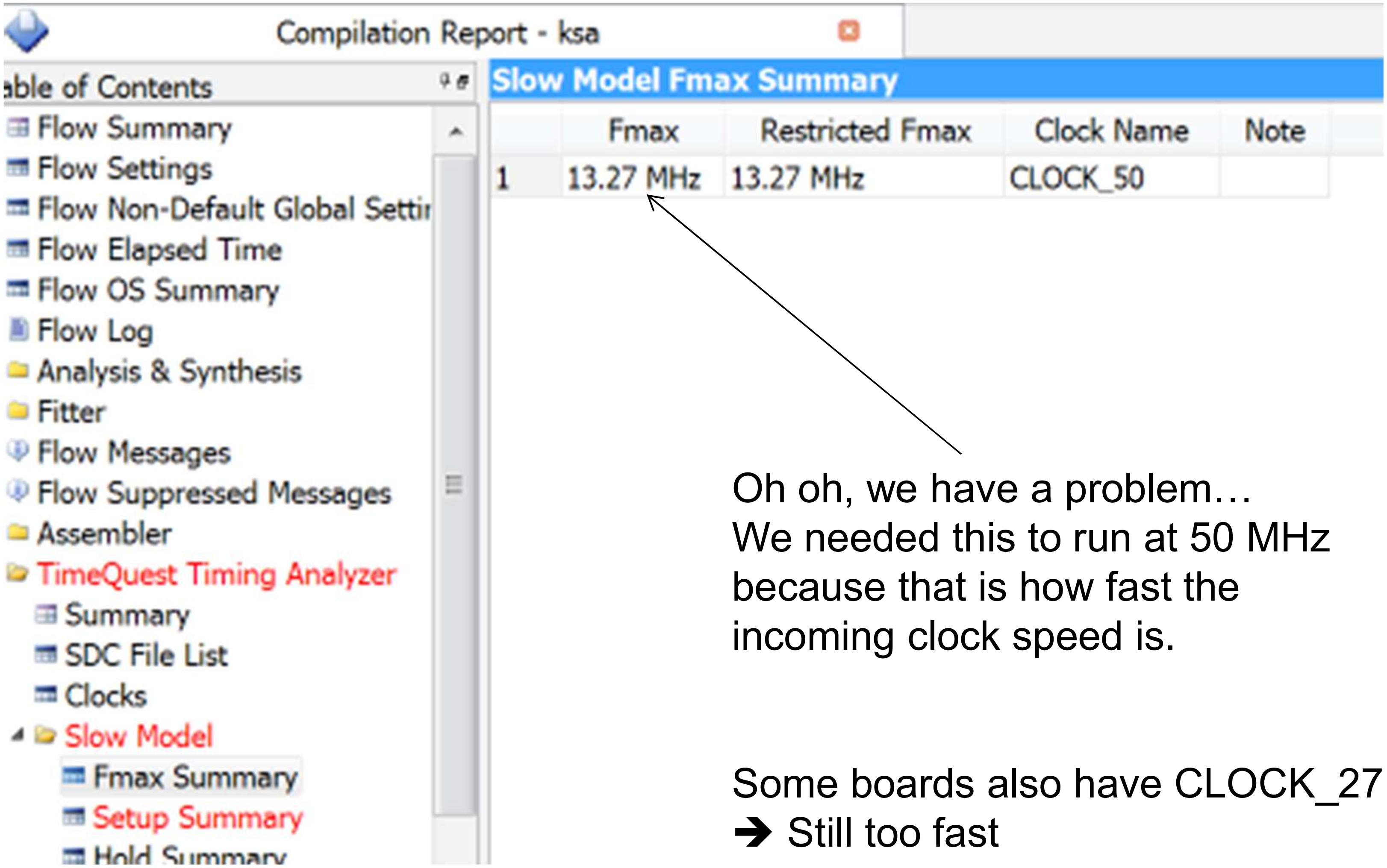
Tradeoffs: Lab 1

Experiments conducted on a DE2 board:

Scenario	Achieved Fmax	Area of Circuit
No explicit timing constraints (1 GHz)	133 MHz	148 Logic Elements
Timing constraint of 130 MHz	130 MHz	143 Logic Elements
Timing constraint of 10 MHz	112 MHz	135 Logic Elements

The next few slides are meant as a kind of “case study” so you understand what timing closure is all about.

Timing Violations



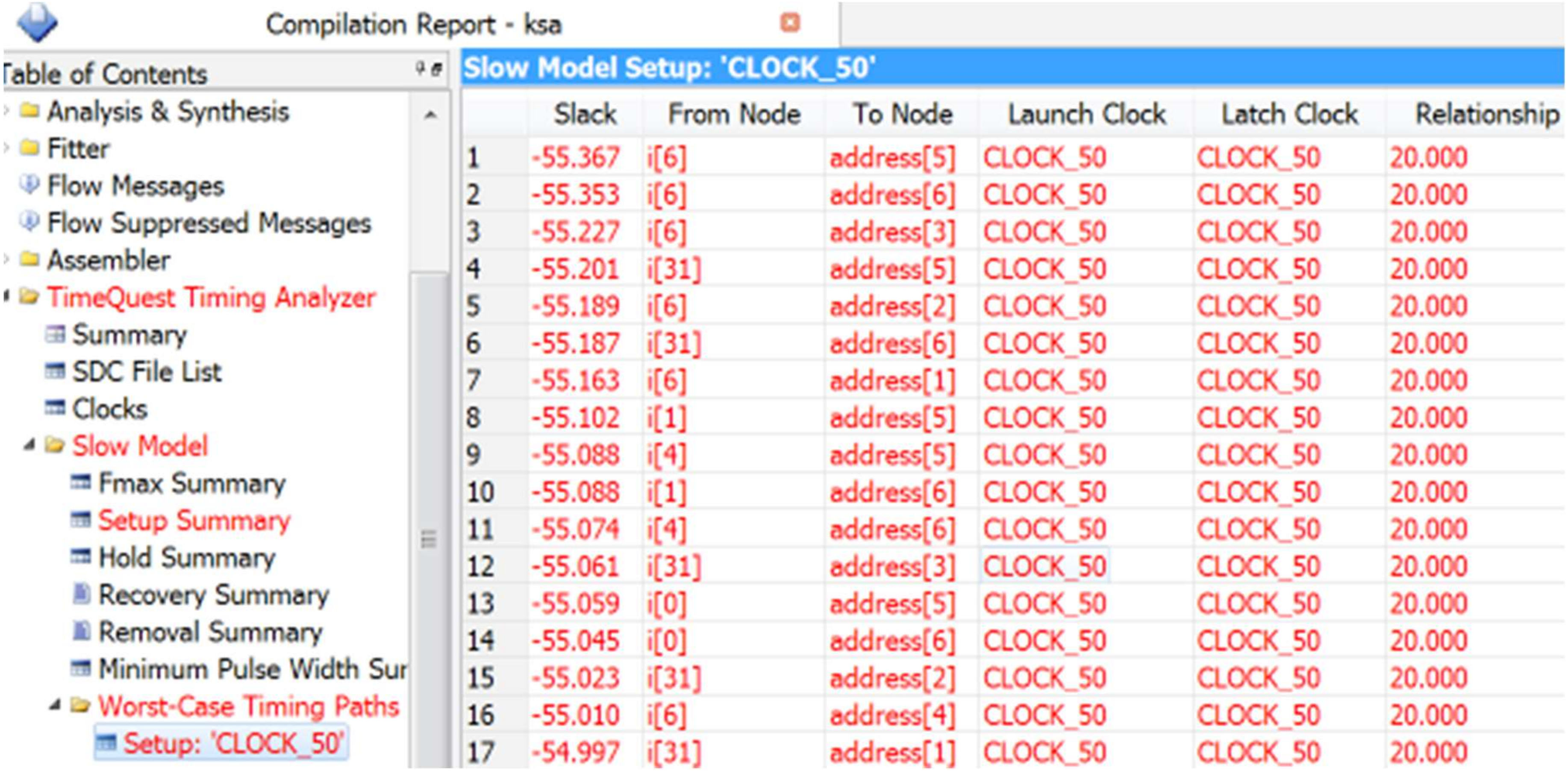
Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	13.27 MHz	13.27 MHz	CLOCK_50	

Oh oh, we have a problem…
We needed this to run at 50 MHz because that is how fast the incoming clock speed is.

Some boards also have CLOCK_27
➔ Still too fast

Let’s investigate: Worst-Case Timing Paths



Slow Model Setup: 'CLOCK_50'

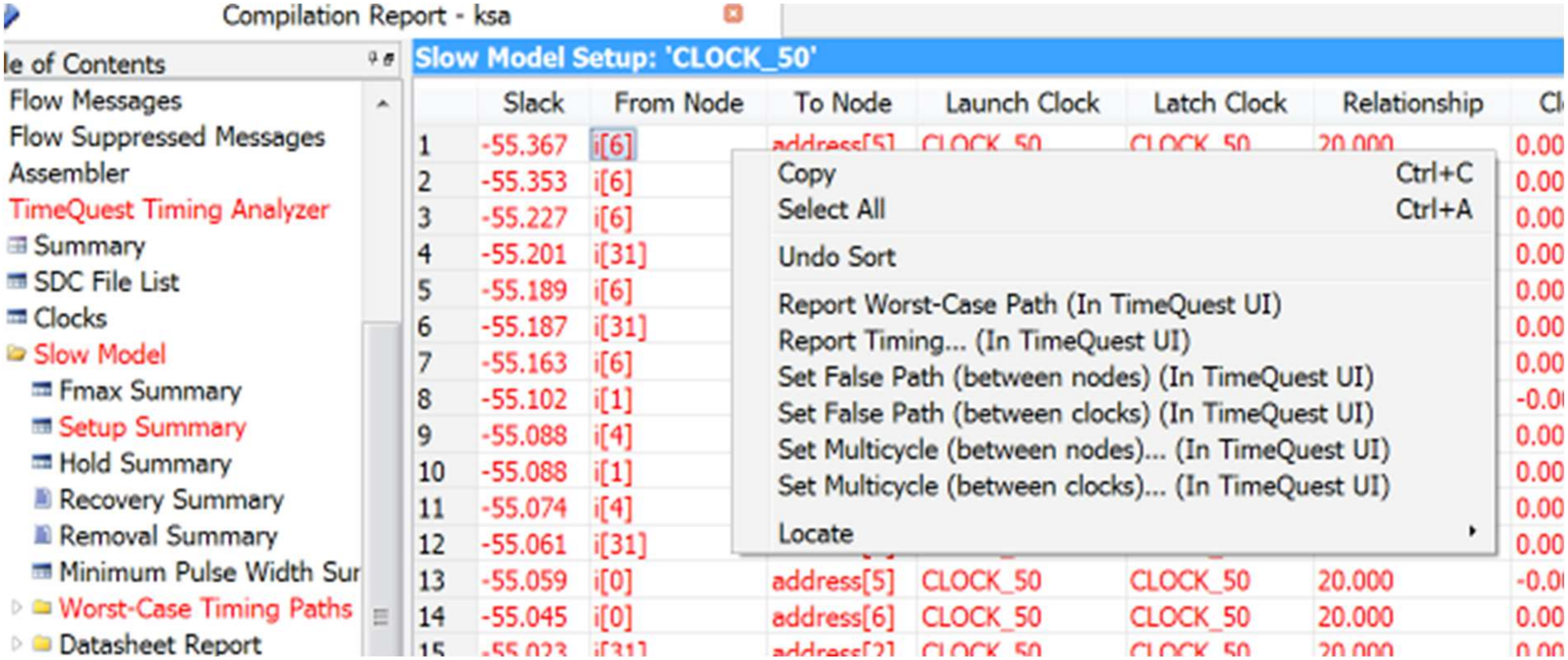
	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship
1	-55.367	i[6]	address[5]	CLOCK_50	CLOCK_50	20.000
2	-55.353	i[6]	address[6]	CLOCK_50	CLOCK_50	20.000
3	-55.227	i[6]	address[3]	CLOCK_50	CLOCK_50	20.000
4	-55.201	i[31]	address[5]	CLOCK_50	CLOCK_50	20.000
5	-55.189	i[6]	address[2]	CLOCK_50	CLOCK_50	20.000
6	-55.187	i[31]	address[6]	CLOCK_50	CLOCK_50	20.000
7	-55.163	i[6]	address[1]	CLOCK_50	CLOCK_50	20.000
8	-55.102	i[1]	address[5]	CLOCK_50	CLOCK_50	20.000
9	-55.088	i[4]	address[5]	CLOCK_50	CLOCK_50	20.000
10	-55.088	i[1]	address[6]	CLOCK_50	CLOCK_50	20.000
11	-55.074	i[4]	address[6]	CLOCK_50	CLOCK_50	20.000
12	-55.061	i[31]	address[3]	CLOCK_50	CLOCK_50	20.000
13	-55.059	i[0]	address[5]	CLOCK_50	CLOCK_50	20.000
14	-55.045	i[0]	address[6]	CLOCK_50	CLOCK_50	20.000
15	-55.023	i[31]	address[2]	CLOCK_50	CLOCK_50	20.000
16	-55.010	i[6]	address[4]	CLOCK_50	CLOCK_50	20.000
17	-54.997	i[31]	address[1]	CLOCK_50	CLOCK_50	20.000

Slowest path is from bit 6 of the i register to bit 5 of the address input of s_mem

“Slack” is difference between “this path delay” and “target clock period” (20ns). Negative slack is BAD. Often hear about “worst-case negative slack” or WCNS

What is this path?

Right click, Report-Worse-Case Path:

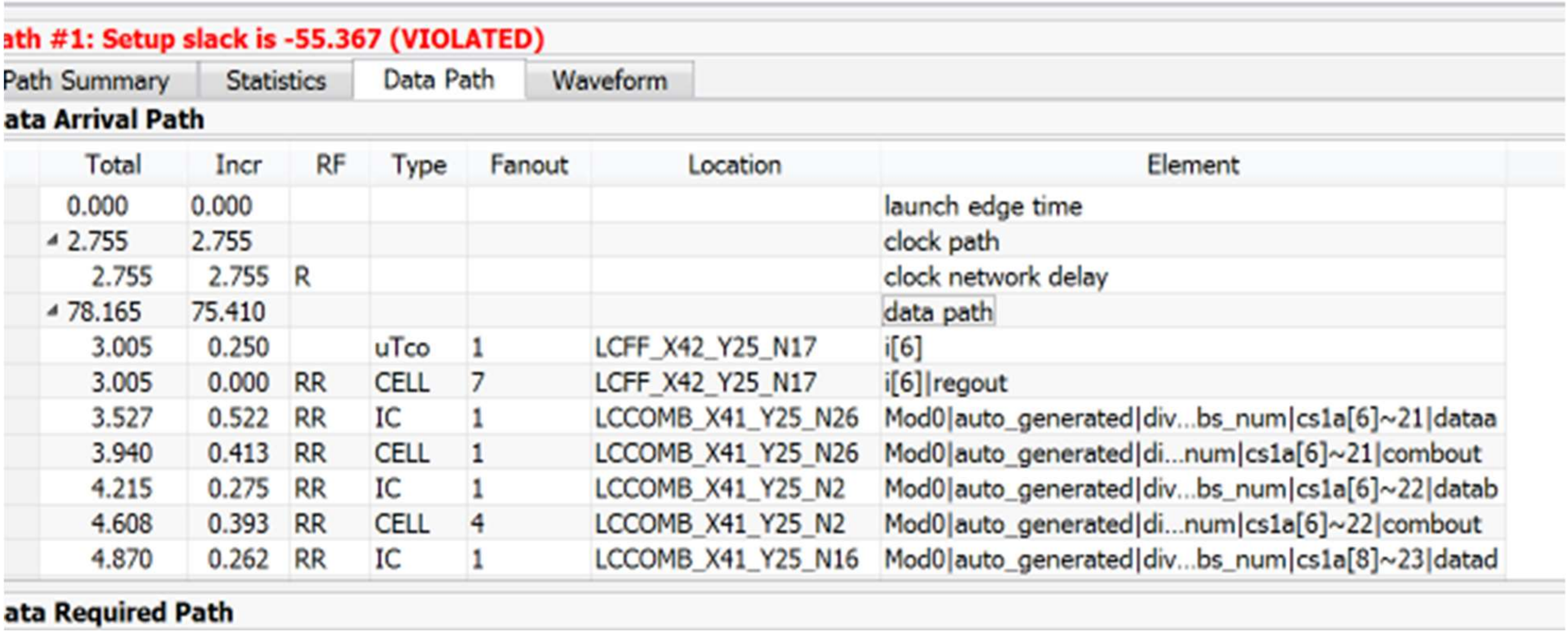

Compilation Report - ksa
le of Contents
Copy Ctrl+C
Select All Ctrl+A
Undo Sort
Report Worst-Case Path (In TimeQuest UI)
Report Timing... (In TimeQuest UI)
Set False Path (between nodes) (In TimeQuest UI)
Set False Path (between clocks) (In TimeQuest UI)
Set Multicycle (between nodes)... (In TimeQuest UI)
Set Multicycle (between clocks)... (In TimeQuest UI)
Locate

Flow Messages
Flow Suppressed Messages
Assembler
TimeQuest Timing Analyzer
Summary
SDC File List
Clocks
Slow Model
Fmax Summary
Setup Summary
Hold Summary
Recovery Summary
Removal Summary
Minimum Pulse Width Sur
Worst-Case Timing Paths
Datasheet Report

Slow Model Setup: 'CLOCK_50'

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Cl
1	-55.367	i[6]	address[5]	CLOCK_50	CLOCK_50	20.000	0.00
2	-55.353	i[6]					0.00
3	-55.227	i[6]					0.00
4	-55.201	i[31]					0.00
5	-55.189	i[6]					0.00
6	-55.187	i[31]					0.00
7	-55.163	i[6]					0.00
8	-55.102	i[1]					-0.0
9	-55.088	i[4]					0.00
10	-55.088	i[1]					0.00
11	-55.074	i[4]					0.00
12	-55.061	i[31]					0.00
13	-55.059	i[0]	address[5]	CLOCK_50	CLOCK_50	20.000	-0.0
14	-55.045	i[0]	address[6]	CLOCK_50	CLOCK_50	20.000	0.00
15	-55.023	i[31]	address[2]	CLOCK_50	CLOCK_50	20.000	0.00

Can go into Timing Quest and select Create Timing Network and then come back…



ath #1: Setup slack is -55.367 (VIOLATED)

ata Arrival Path

Total	Incr	RF	Type	Fanout	Location	Element
0.000	0.000					launch edge time
2.755	2.755					clock path
2.755	2.755	R				clock network delay
78.165	75.410					data path
3.005	0.250		uTco	1	LCFF_X42_Y25_N17	i[6]
3.005	0.000	RR	CELL	7	LCFF_X42_Y25_N17	i[6]|regout
3.527	0.522	RR	IC	1	LCCOMB_X41_Y25_N26	Mod0|auto_generated|div...bs_num|cs1a[6]~21|dataa
3.940	0.413	RR	CELL	1	LCCOMB_X41_Y25_N26	Mod0|auto_generated|di...num|cs1a[6]~21|combout
4.215	0.275	RR	IC	1	LCCOMB_X41_Y25_N2	Mod0|auto_generated|div...bs_num|cs1a[6]~22|datab
4.608	0.393	RR	CELL	4	LCCOMB_X41_Y25_N2	Mod0|auto_generated|di...num|cs1a[6]~22|combout
4.870	0.262	RR	IC	1	LCCOMB_X41_Y25_N16	Mod0|auto_generated|div...bs_num|cs1a[8]~23|datad

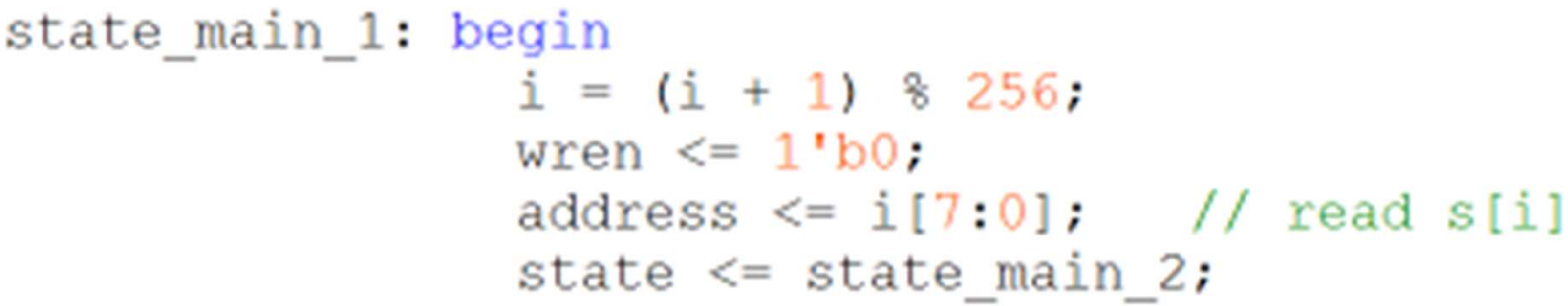
ata Required Path

It is coming out of register i and going to a divider...

Combinational dividers are notoriously slow…

Look at the code to see what is happening…

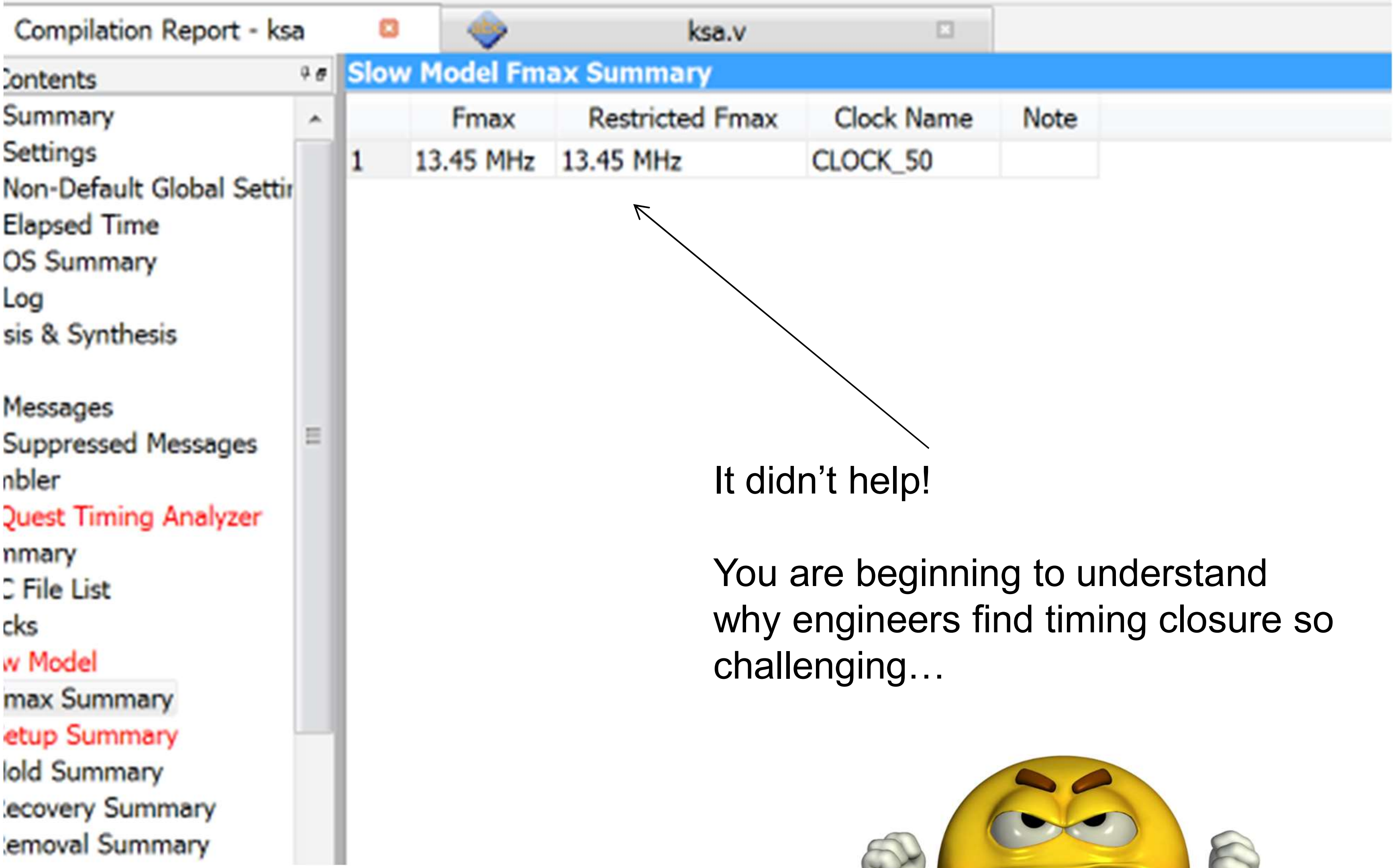
In my code, I found something like:



```
state_main_1: begin
                i = (i + 1) % 256;
                wren <= 1'b0;
                address <= i[7:0];   // read s[i]
                state <= state_main_2;
```

I suspect the division might be due to this mod function. But, if course, mod 256 is really just taking the 8 order LSB. We could get rid of that and just remove the % 256.

So, I do it and recompile…



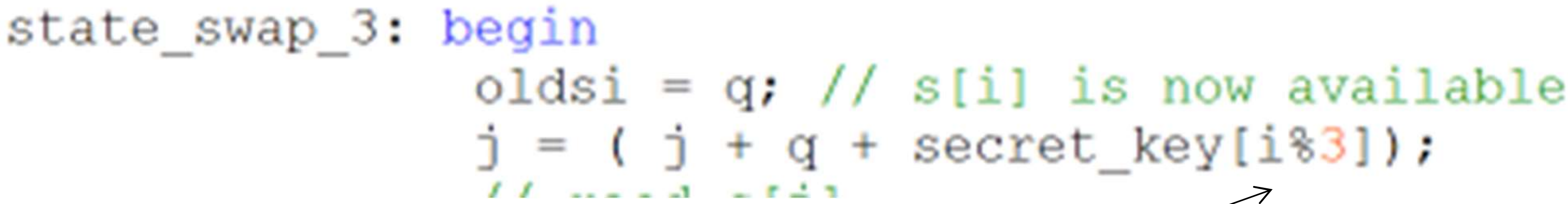
Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	13.45 MHz	13.45 MHz	CLOCK_50	

It didn't help!

You are beginning to understand why engineers find timing closure so challenging…

Going back to my code, I see:

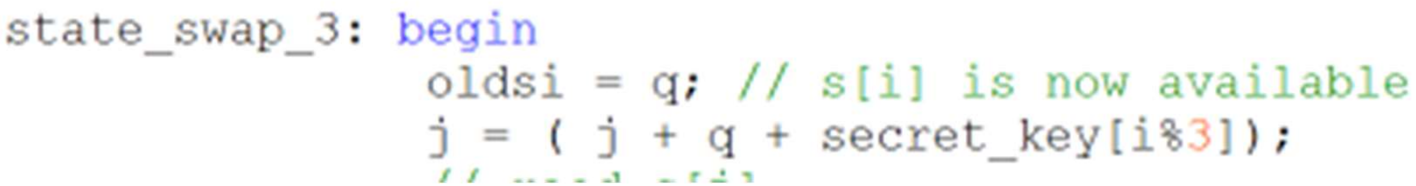


```
state_swap_3: begin
                oldsi = q; // s[i] is now available
                j = ( j + q + secret_key[i%3]);
```

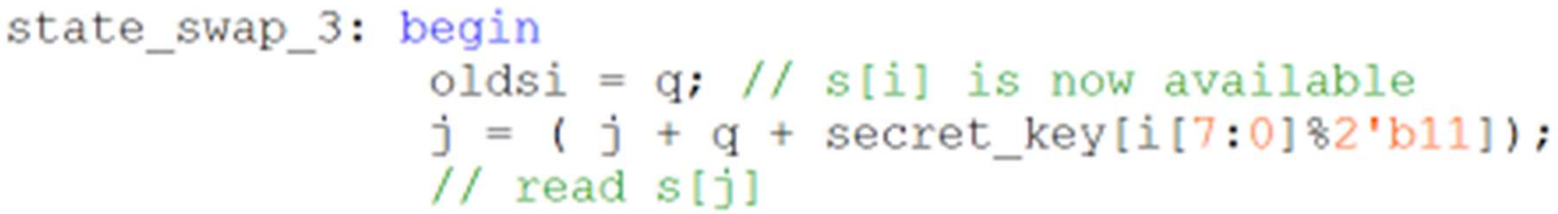
Certainly if we are dividing by 3, this should be a small enough circuit right?

But wait…. in my implementation, i is an integer (32 bits) and the constant 3 is interpreted as 32 bits. So this constructs a 32 bit divider!

So, replace it with this:

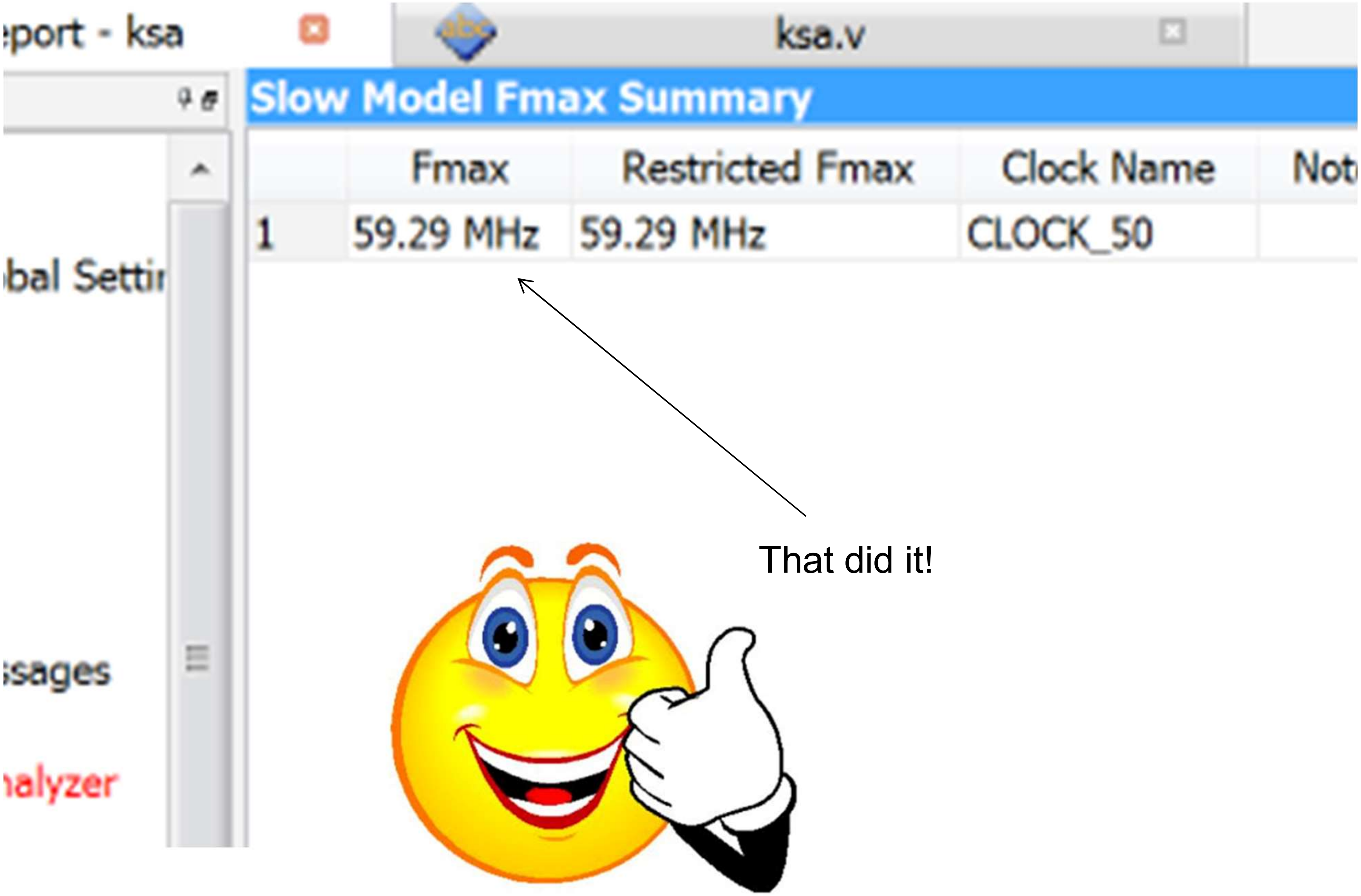


```
state_swap_3: begin
                oldsi = q; // s[i] is now available
                j = ( j + q + secret_key[i%3]);
                // read s[j]
```



```
state_swap_3: begin
                oldsi = q; // s[i] is now available
                j = ( j + q + secret_key[i[7:0]%2'b11]);
                // read s[j]
```

Then recompile and …



Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Not
1	59.29 MHz	59.29 MHz	CLOCK_50	

That did it!

That was a case study to introduce you to the ~~frustrations~~ concept of timing closure.

Can Go Faster Than Predicted Fmax?

Overclocking

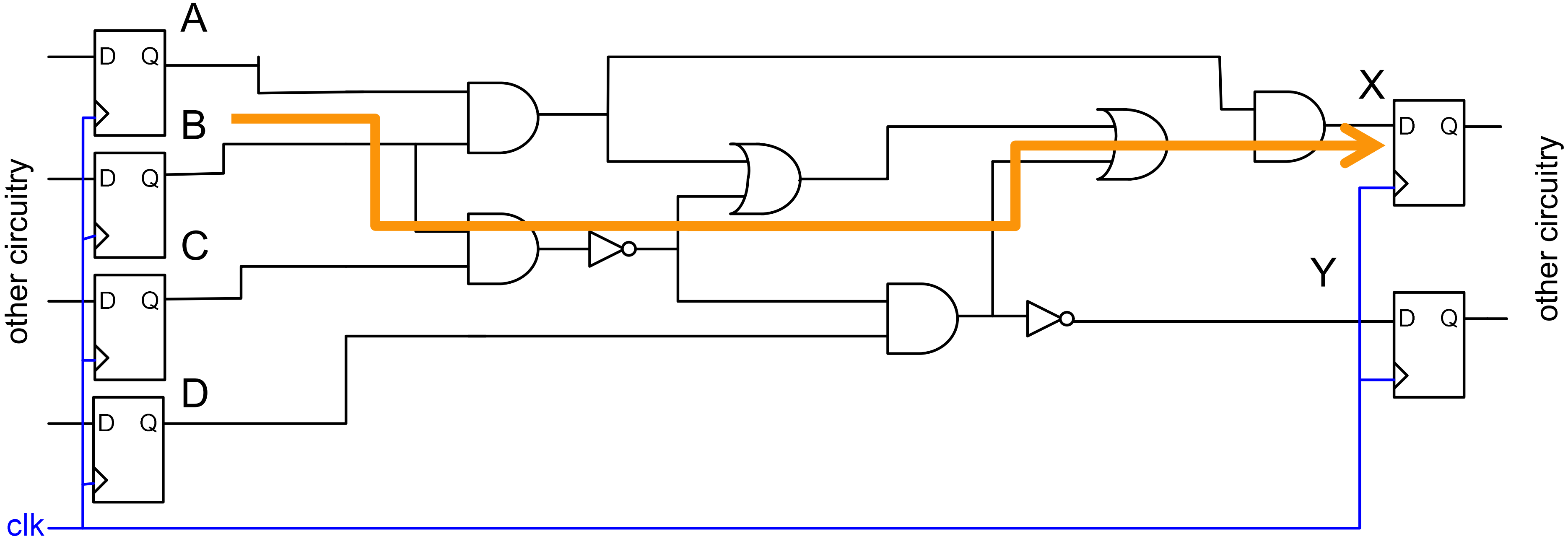
- Quartus gives conservative estimate
- Actual delays vary chip-to-chip

Dangerous… don’t do it

- In testing, it might appear to work. But, how do you know that you are exercising the critical path?
- Some chips will be slower than the one used for prototyping
- Conditions such as temperature can affect gate delays

PIPELINING

From earlier: Critical Path Delay

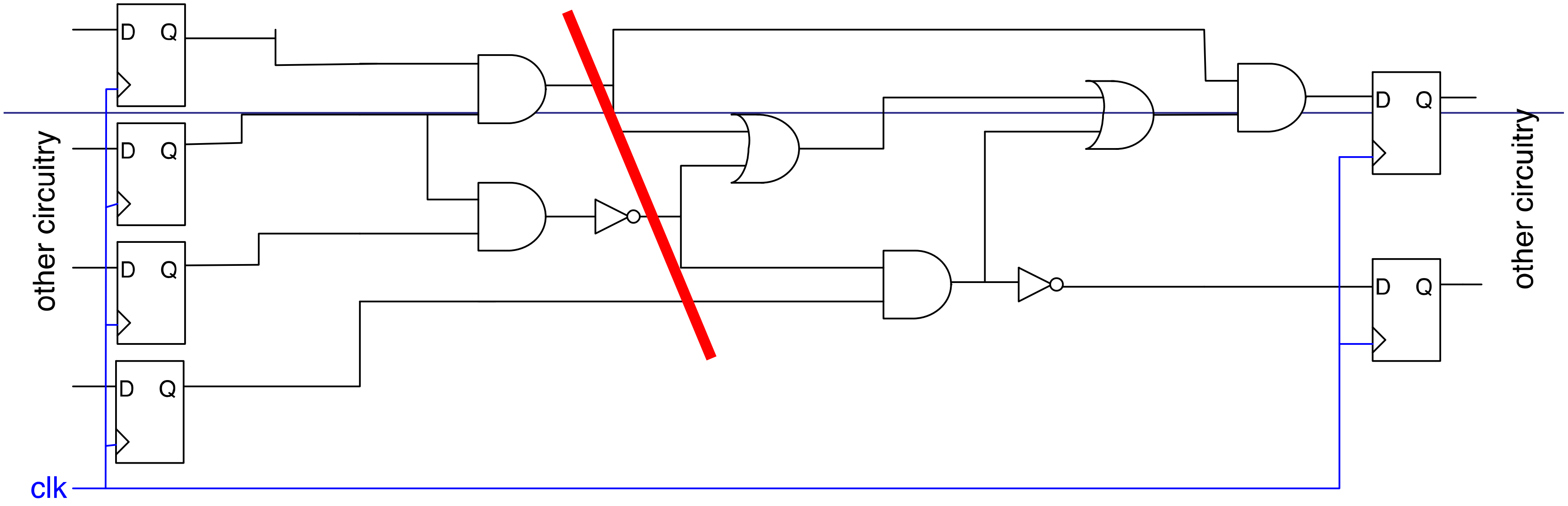


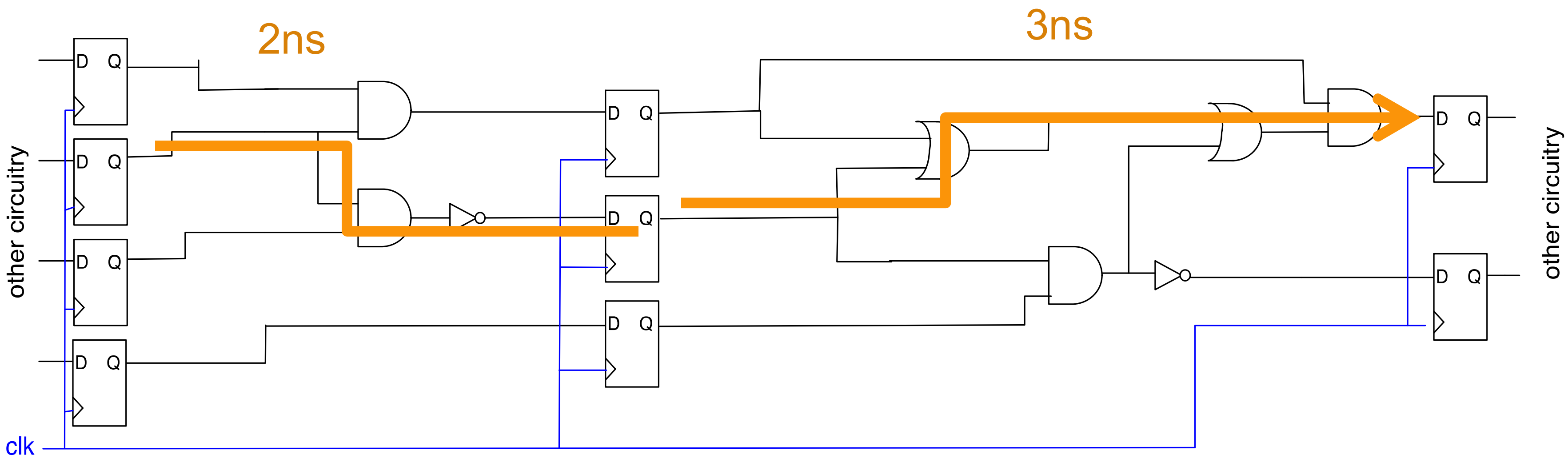
Critical Path is

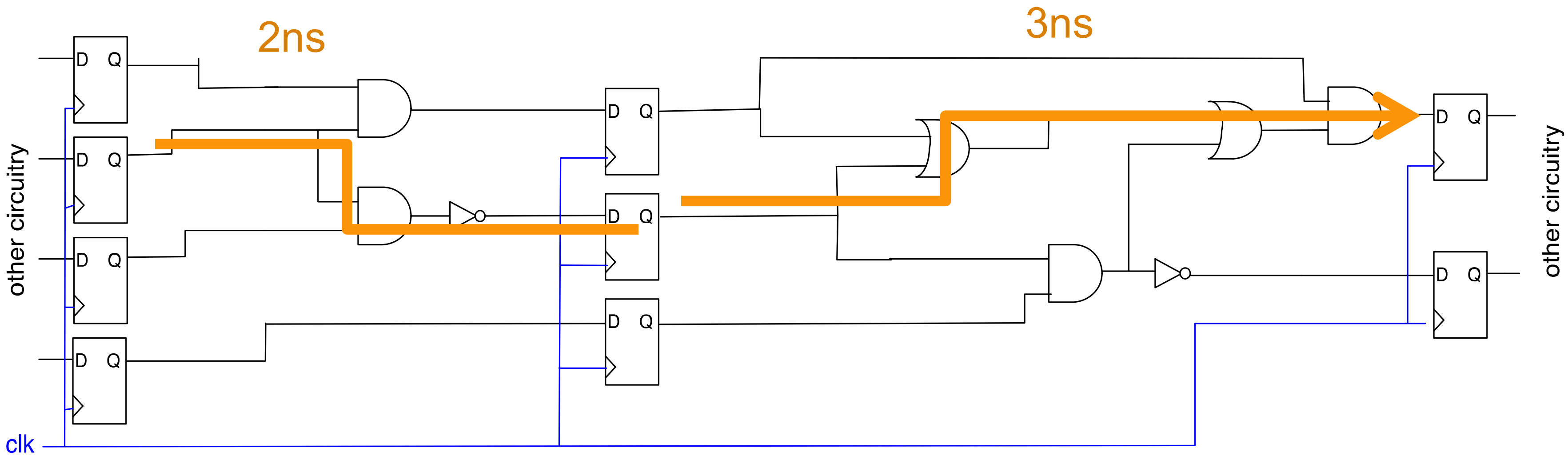
B → AND → INV → OR → OR → AND → X

Critical Path Delay is 5ns

Clock period cannot be smaller than 5ns or else X register will read in wrong data


D
Q
D
Q
D
Q
D
Q
D
Q
D
Q
other circuitry
other circuitry
clk


2ns
3ns
D
Q
D
Q
D
Q
D
Q
D
Q
D
Q
D
Q
D
Q
D
Q
other circuitry
other circuitry
clk

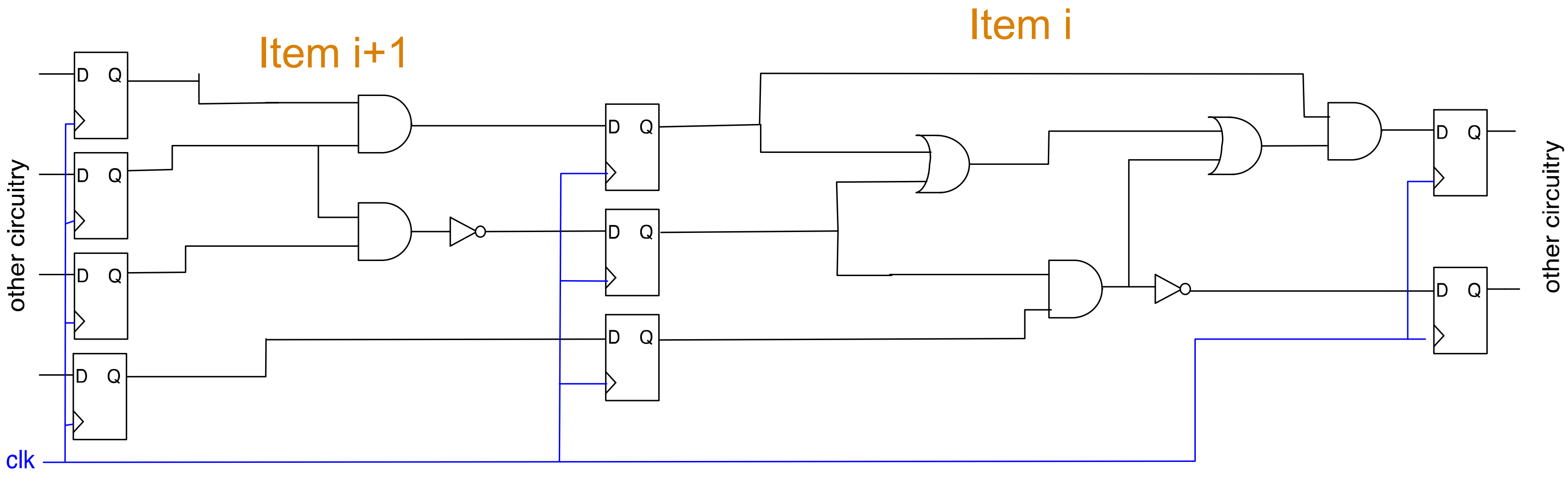


The new longest path between registers is 3 ns.
So, this now goes at 1/3ns = 333 MHz (before it was 1/5 = 200 MHz)

But it takes 2 cycles to get a result now.
Have we really improved anything?

Pipelining might help if:

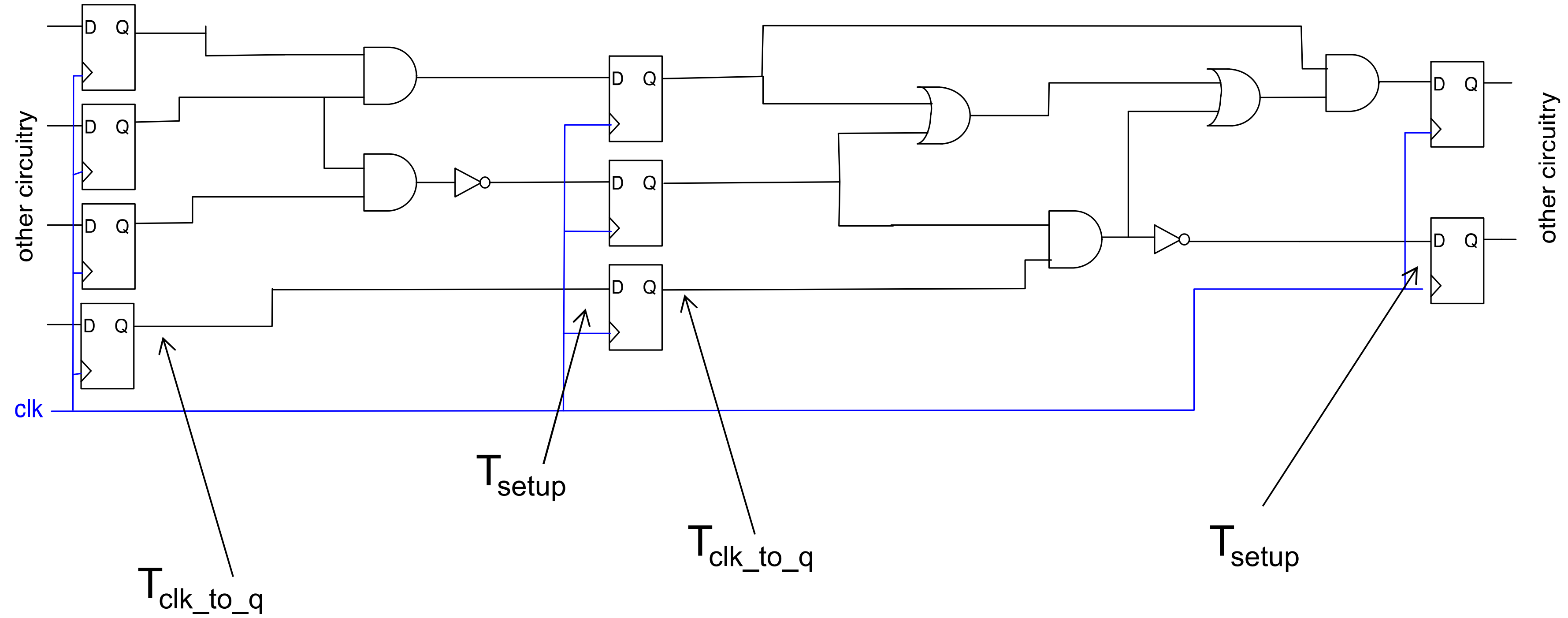
1. One (or several) of the paths are very long, compared to the others. In that case, these slow paths dictate performance. If we pipeline the slow parts, we can run the whole circuit faster.
2. We can sometimes have multiple data items "in flight" at once:



Limits to pipelining

There is a limit to how much benefit you can get from pipelining:

- Every pipeline stage has the overhead of $t_{clk\_to\_q}$ and t_{setup}.



In an FPGA it can be even worse.

In some FPGAs, signal must pass through extra interconnect and a LUT just to use the flip-flop.

Pipelining changes the timing behaviour of your circuit

- Need to take this into account when you are designing your datapath + controller

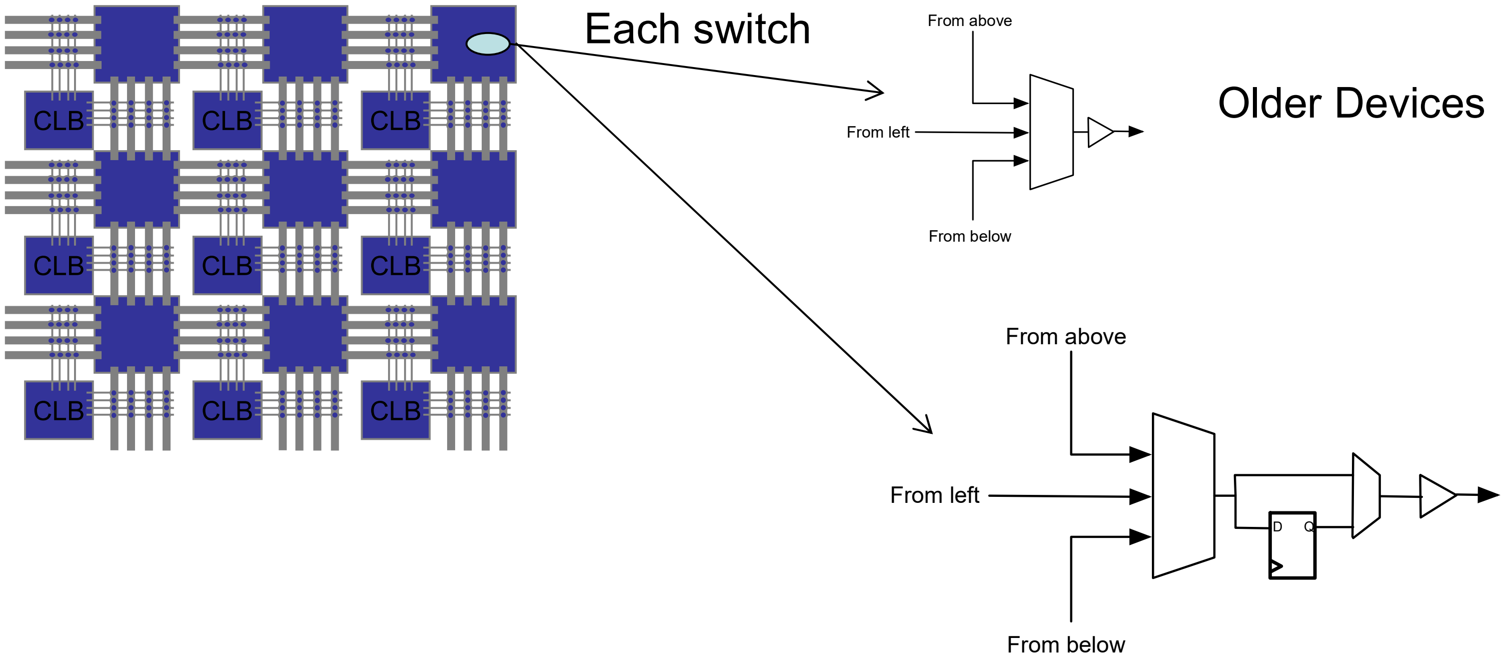
Quartus II will **not** pipeline your circuit for you. Why?

- Synthesis tools are “cycle accurate”: the behaviour of the circuit must be the same, cycle-by-cycle, as the Verilog specification.
- Pipelining a design would create a different cycle-by-cycle behaviour

Hyper-Pipelining in Stratix 10

Most recent device family has an optional flip-flop at *every switch* in the routing.

NOTE: can’t use reset on these FFs (design should avoid use of resets)



Each switch

Older Devices

Devices supporting
Hyper-Pipelining (Stratix 10)

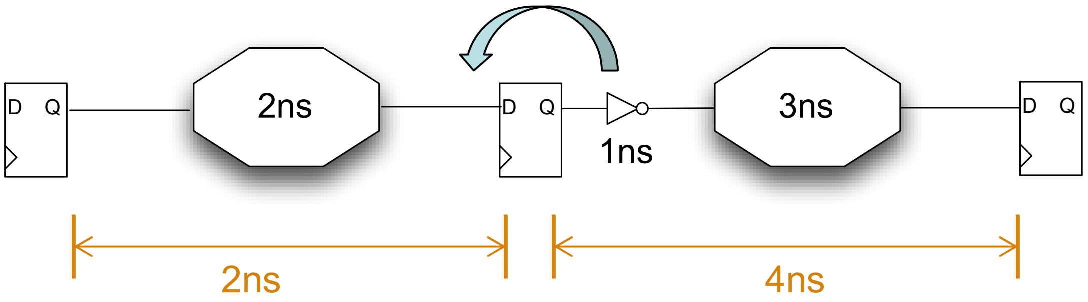
Important Point: Modern FPGAs support very fast clock speeds and heavily pipelined architectures. Lots of registers.

RETIMING

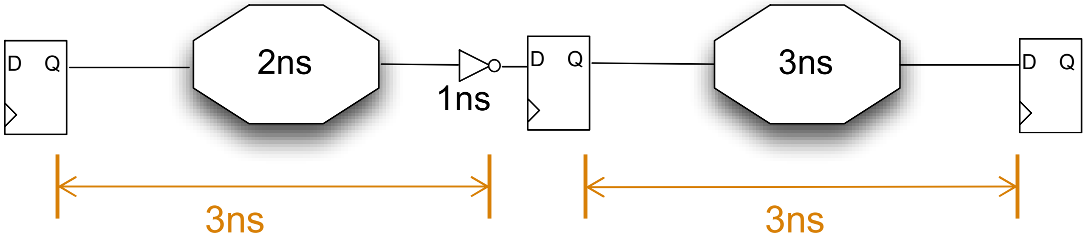
Retiming

Key to pipelining: make every stage balanced (same delay)

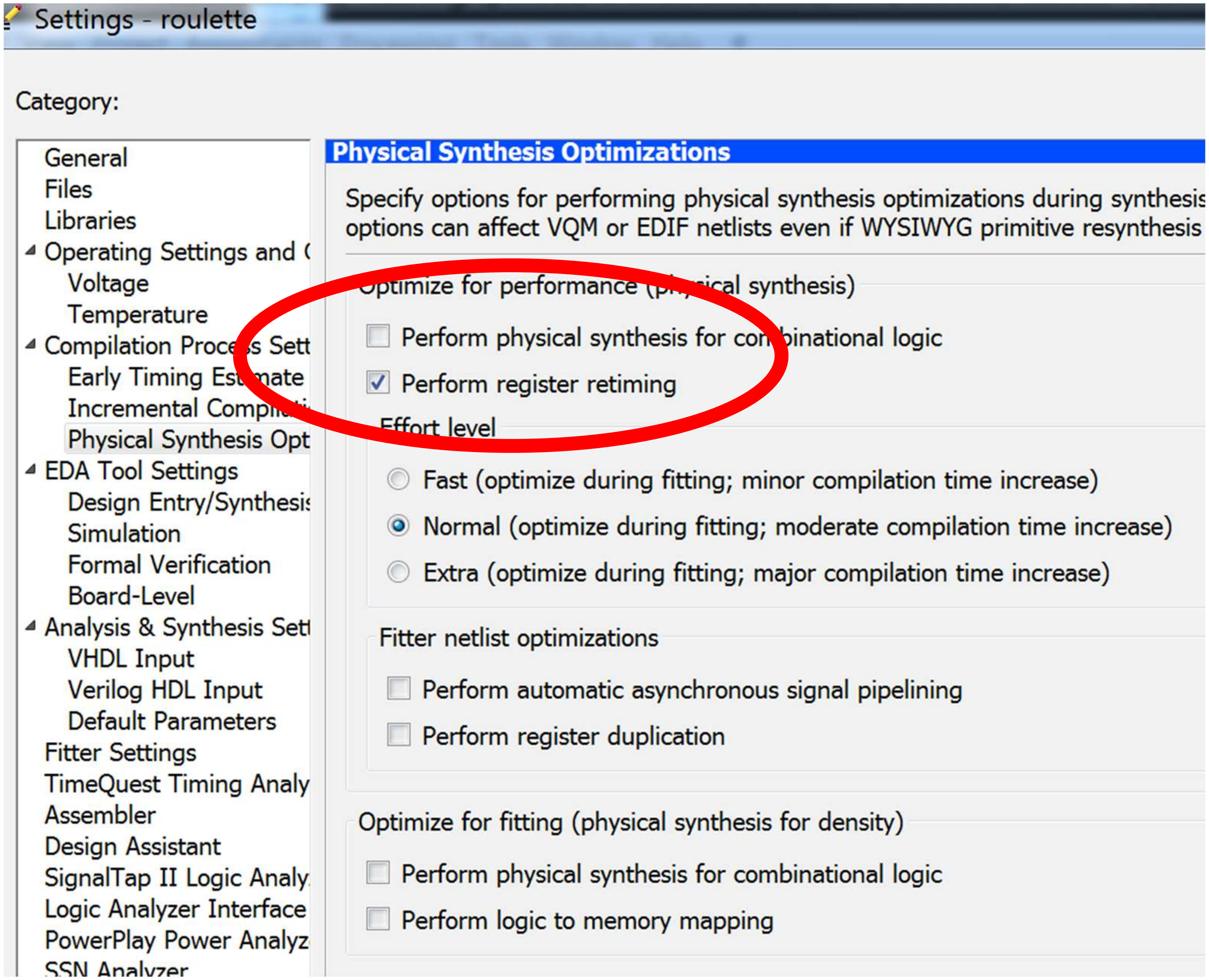
To balance, may need to move gates:



Max Freq =
1/4ns =
250 MHz



Max Freq =
1/3ns =
333 Mhz



Settings - roulette

Category:

- General
- Files
- Libraries
- Operating Settings and (
  - Voltage
  - Temperature
- Compilation Process Sett
  - Early Timing Estimate
  - Incremental Compilati
  - Physical Synthesis Opt
- EDA Tool Settings
  - Design Entry/Synthesis
  - Simulation
  - Formal Verification
  - Board-Level
- Analysis & Synthesis Sett
  - VHDL Input
  - Verilog HDL Input
  - Default Parameters
- Fitter Settings
- TimeQuest Timing Analy
- Assembler
- Design Assistant
- SignalTap II Logic Analy
- Logic Analyzer Interface
- PowerPlay Power Analyz
- SSN Analyzer

Physical Synthesis Optimizations

Specify options for performing physical synthesis optimizations during synthesis
options can affect VQM or EDIF netlists even if WYSIWYG primitive resynthesis

Optimize for performance (physical synthesis)

☐ Perform physical synthesis for combinational logic

☑ Perform register retiming

Effort level

○ Fast (optimize during fitting; minor compilation time increase)

● Normal (optimize during fitting; moderate compilation time increase)

○ Extra (optimize during fitting; major compilation time increase)

Fitter netlist optimizations

☐ Perform automatic asynchronous signal pipelining

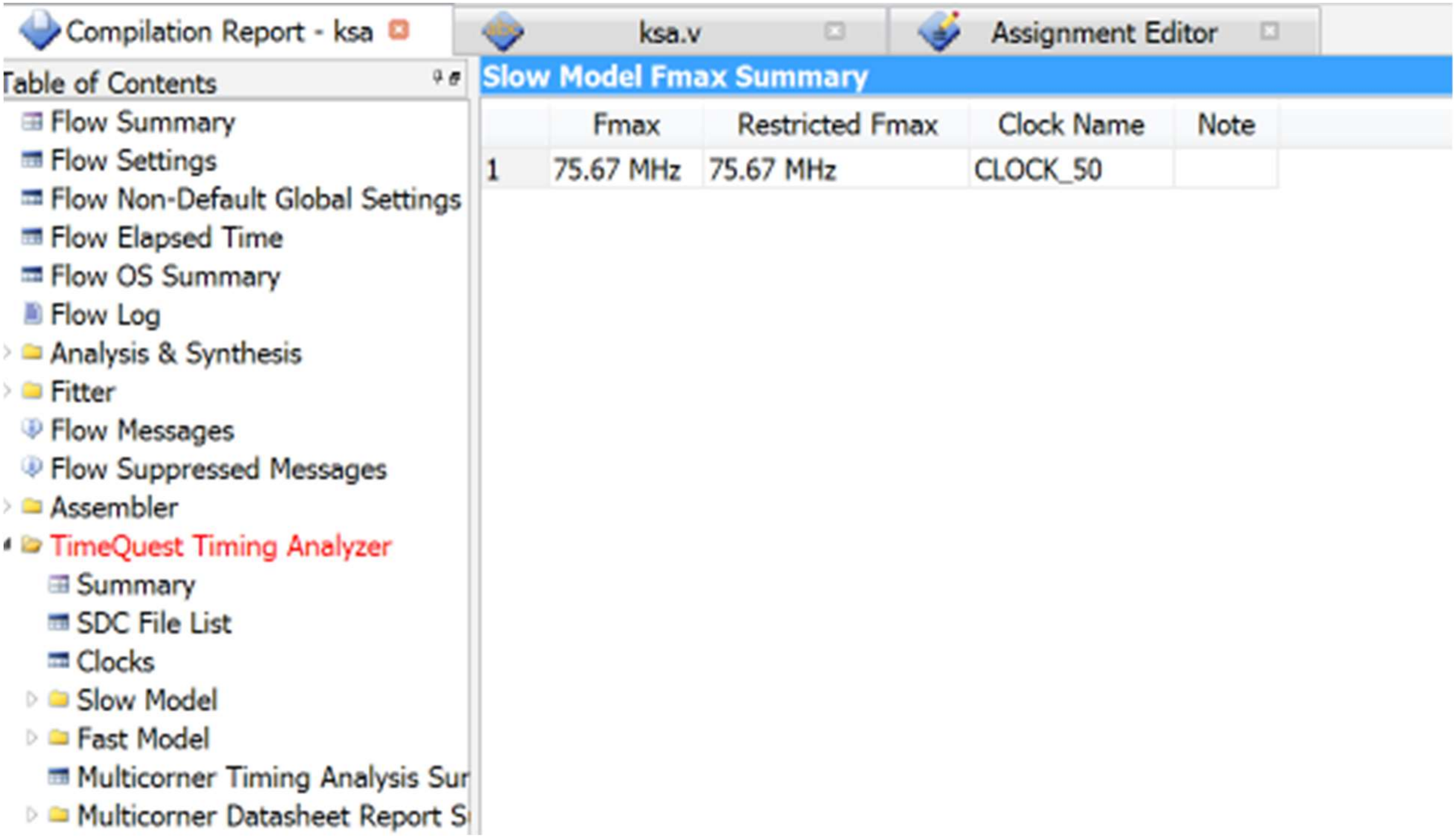
☐ Perform register duplication

Optimize for fitting (physical synthesis for density)

☐ Perform physical synthesis for combinational logic

☐ Perform logic to memory mapping

This increased my Fmax from 59 MHz to 76 MHz !!
(…not always successful…)

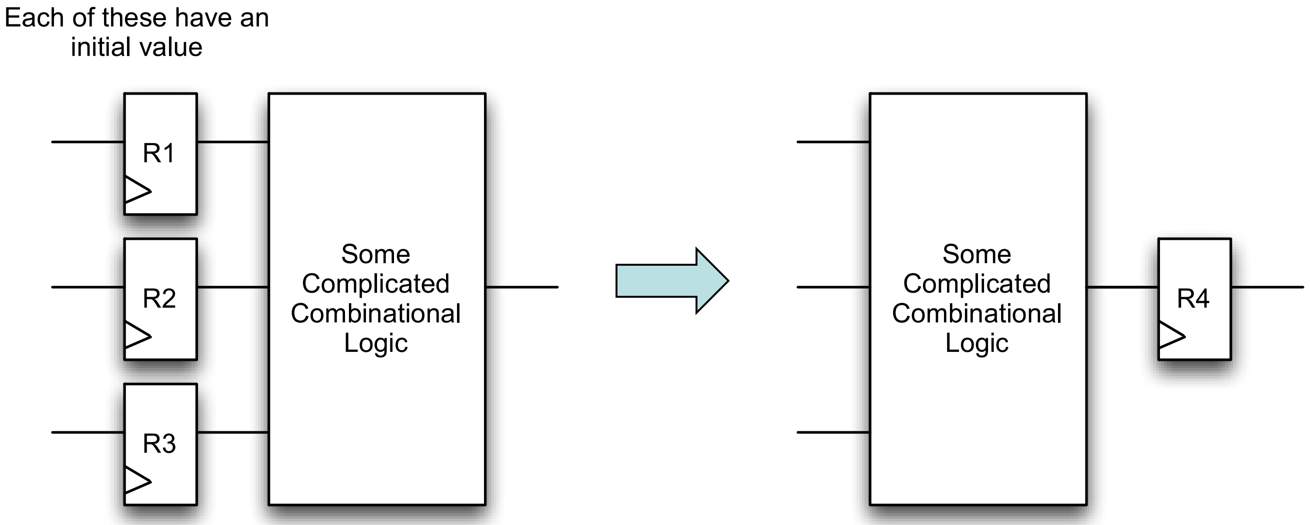


Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	75.67 MHz	75.67 MHz	CLOCK_50	

Initial Value Problem with Re-timing:

Forward Re-Timing:


Each of these have an initial value
R1
R2
R3
Some Complicated Combinational Logic
Some Complicated Combinational Logic
R4

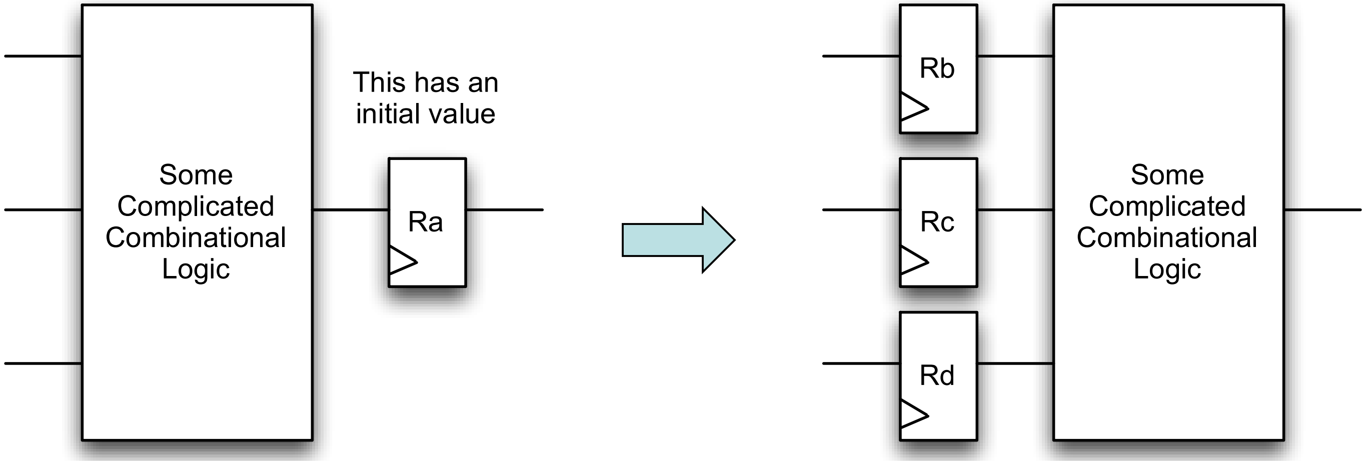
What is the initial value of R4?

Can Quartus determine it automatically?

Yes, eg simulate combinational logic with initial values of R1,R2,R3.

Initial Value Problem with Re-timing:

Backward Re-Timing:


Some Complicated Combinational Logic
This has an initial value
Ra
Rb
Rc
Rd
Some Complicated Combinational Logic

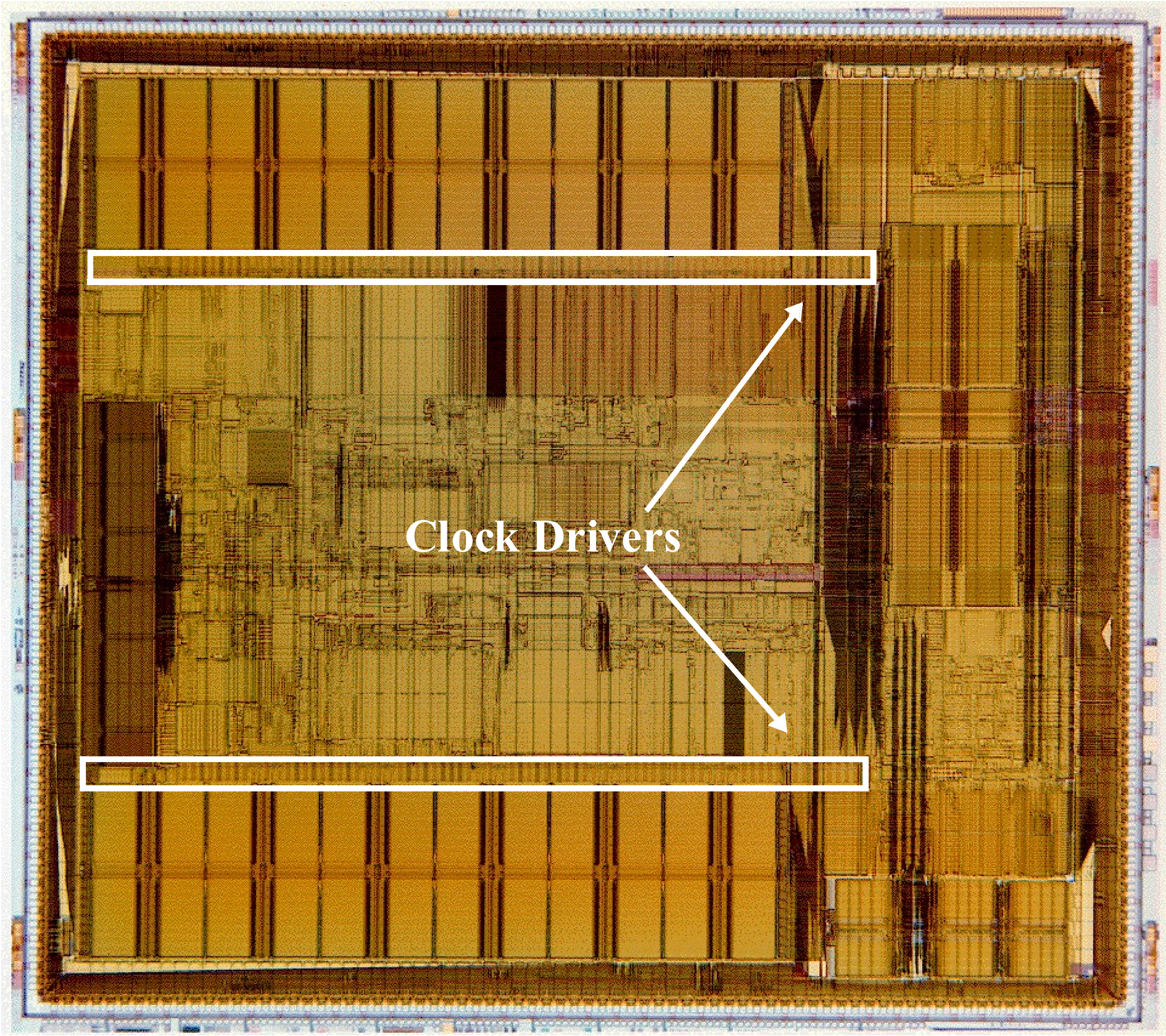
What is initial value of Rb, Rc, and Rd?

(given the initial value of Ra specified by the user)

No. Simulation doesn’t work “backwards”. Must go through all combinations of Rb, Rc, and Rd to determine which combination gives Ra.

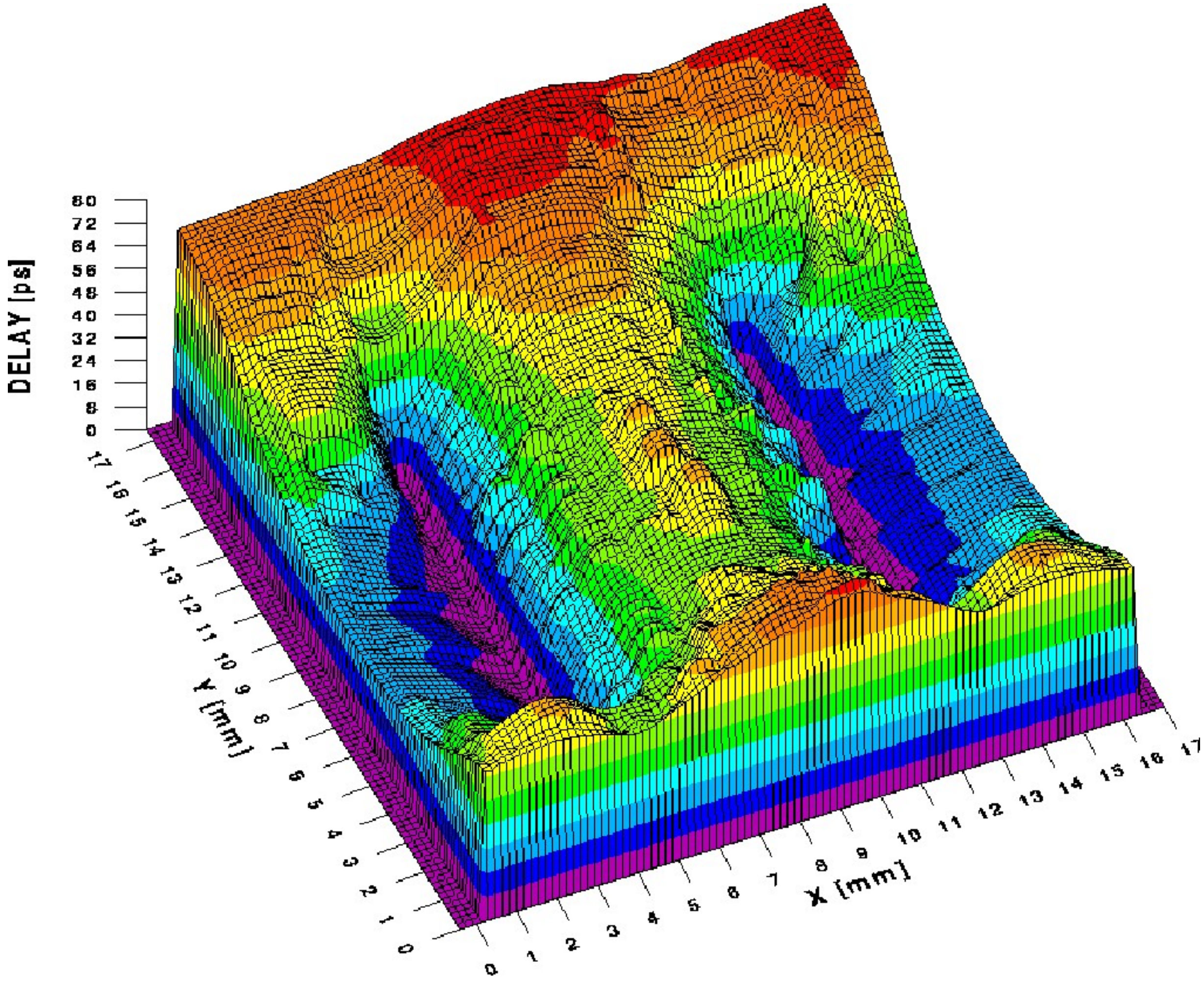
So: Initial values for registers may limit opportunities for retiming.

Clock Skew



Alpha 21164 (EV5)

Clock Skew in Alpha Processor



© Prentice Hall

Clock Skew

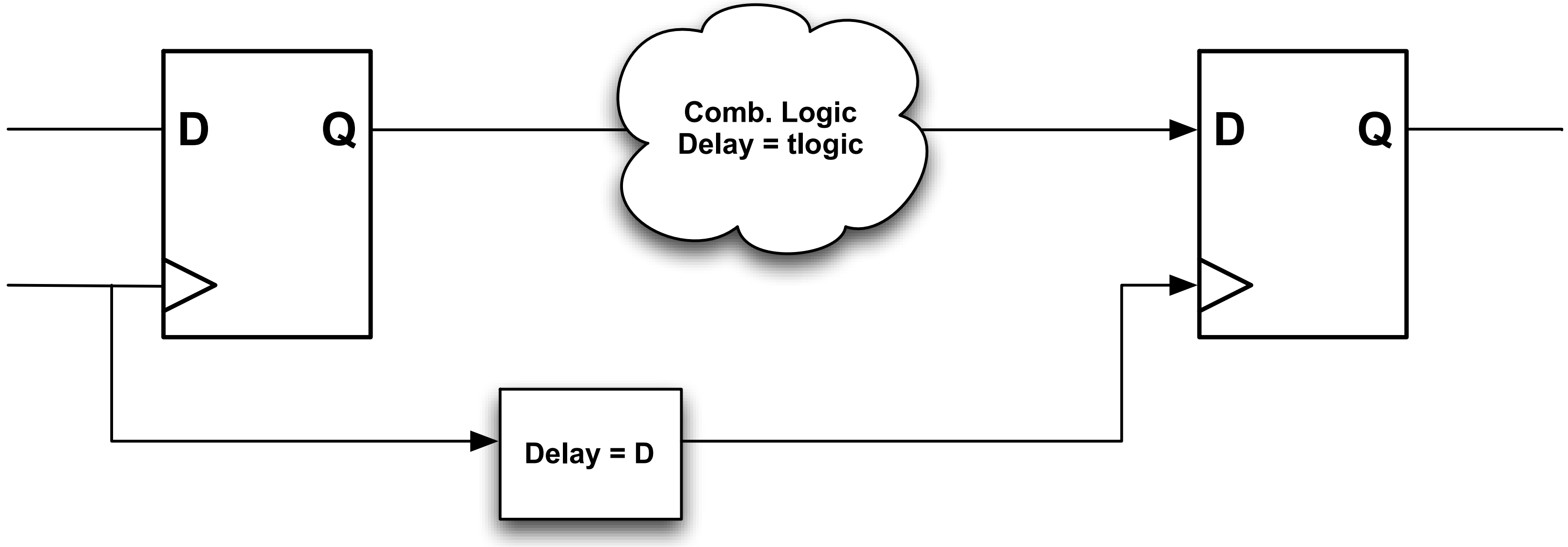
Clock Skew is very real:

- We can not guarantee that the clock edge arrives at all flip-flops at the same time

Implications:

- Improvement in Fmax (very unlikely)
- Reduction in Fmax
- Failure of the design, regardless of Fmax

Clock Skew



If D is 0 (no skew), we know from the previous slide set that:

$$t_{\text{clock}} >= t_{\text{clk\_to\_q}} + t_{\text{logic}} + t_{\text{setup}}$$

What if D > 0 (clock skew)?

$$t_{\text{clock}} + D >= t_{\text{clk\_to\_q}} + t_{\text{logic}} + t_{\text{setup}}$$

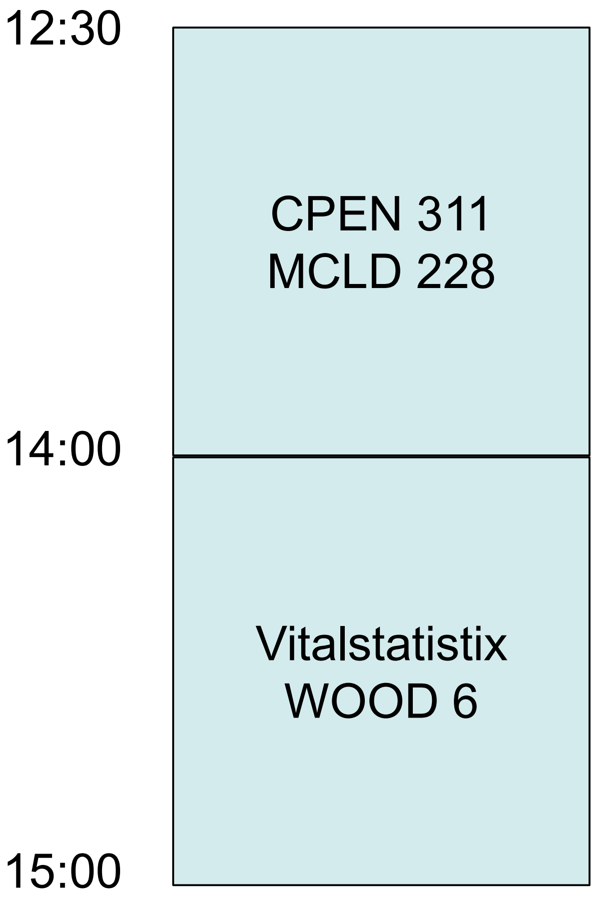
Or:

$$t_{\text{clock}} >= t_{\text{clk\_to\_q}} + t_{\text{logic}} + t_{\text{setup}} - D$$

In this case, clock skew increased our clock frequency!

Analogy

Suppose this is your course schedule:



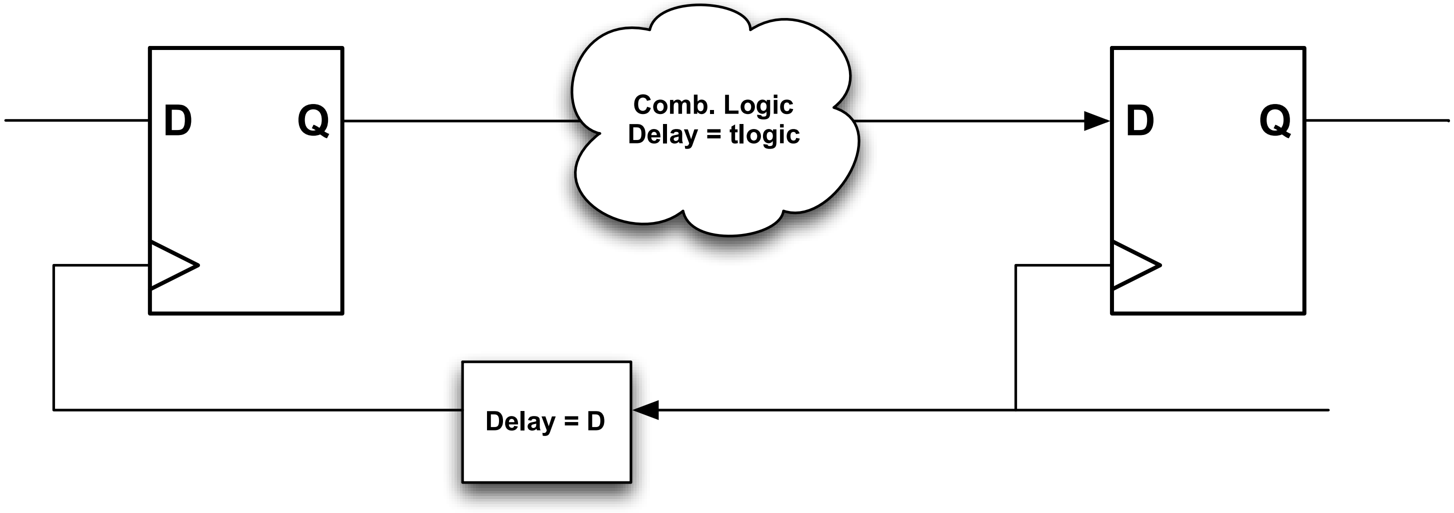
Nominally, you have **10 minutes** to get from one class to the other.

But, if next prof's watch is 2 minutes late, you have **12 minutes** to get there.

What if next prof's watch is 2 minutes early?

You've only got **8 minutes**!

Clock Skew

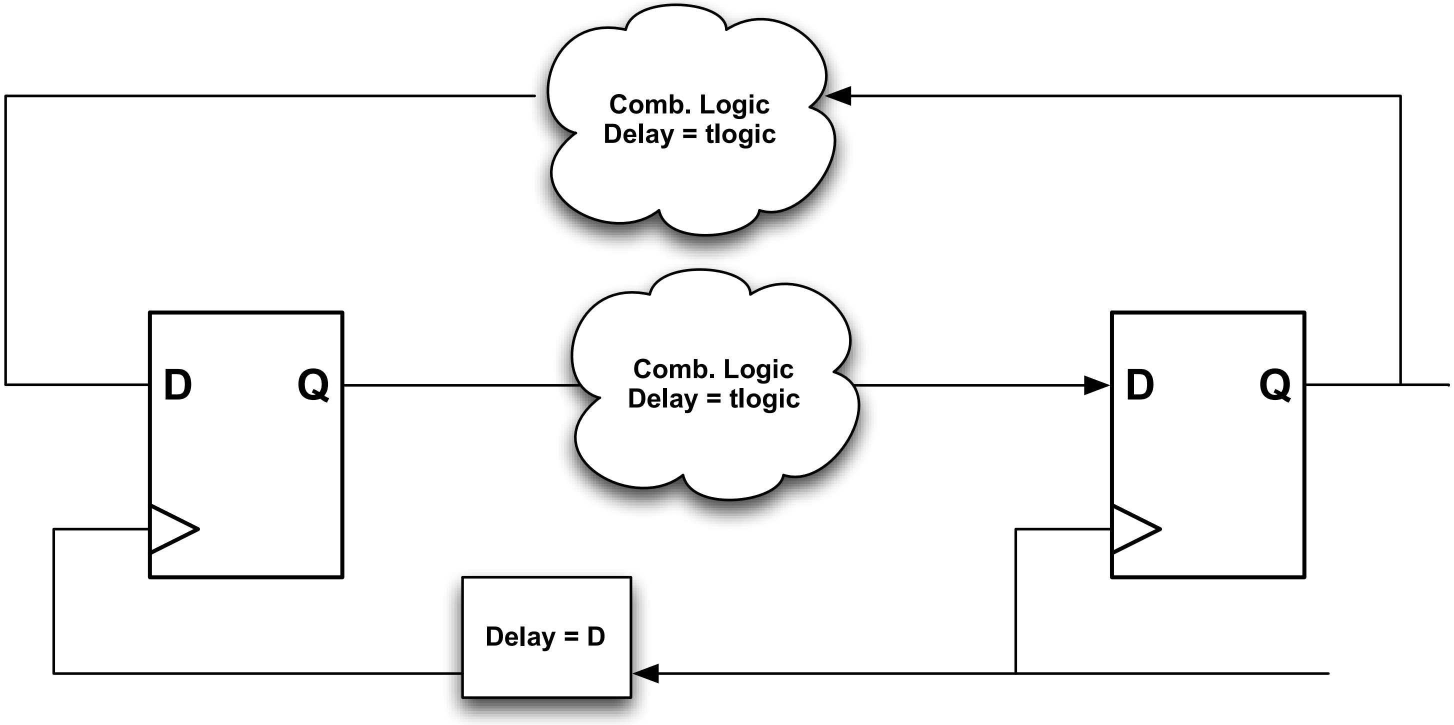


$t_{clock} - D >= t_{clk\_to\_q} + t_{logic} + t_{setup}$

Or:

$t_{clock} >= t_{clk\_to\_q} + t_{logic} + t_{setup} + D$

In this case, clock skew decreased our clock frequency!



In this case, the minimum clock period:

- decreases due to top path
- increases due to bottom path

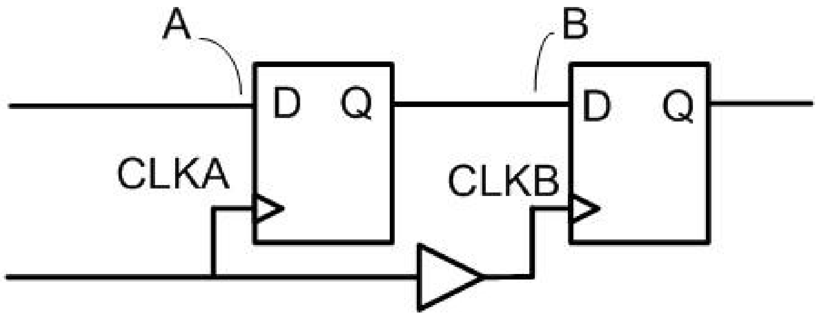
Since the critical path is the worst case path:

$$t_{\text{clock}} >= t_{\text{clk\_to\_q}} + t_{\text{logic}} + t_{\text{setup}} + D$$

ie. Overall, the clock skew increases the critical path (slows us down)

Clock Skew

Clock skew can also cause hold time violations:



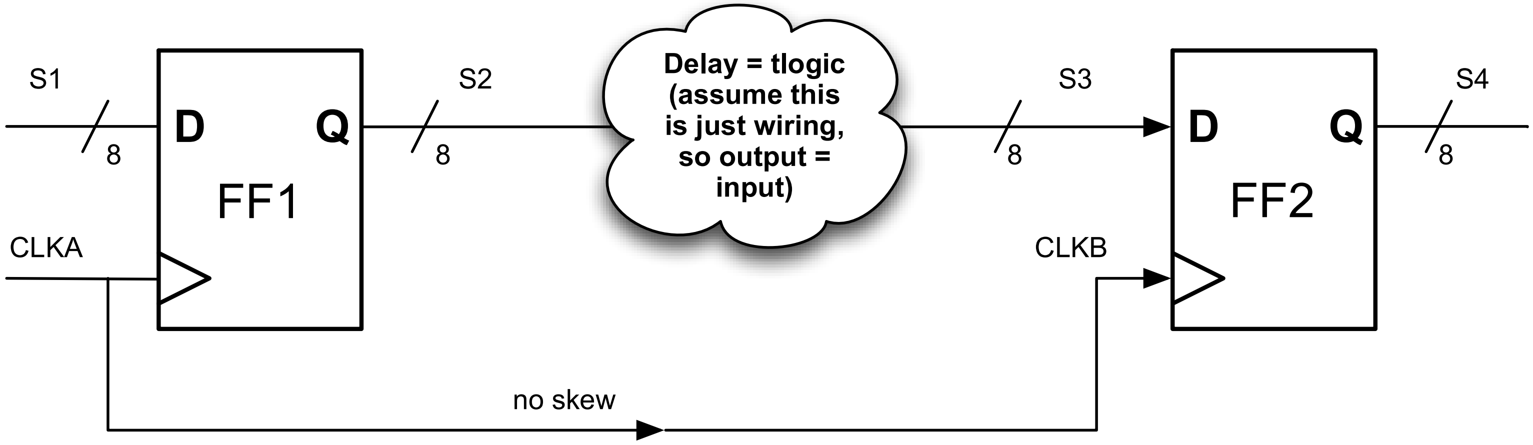
Tool needs to check for the worst case skew when checking hold time violations.

To fix, it increases routing delay to B.

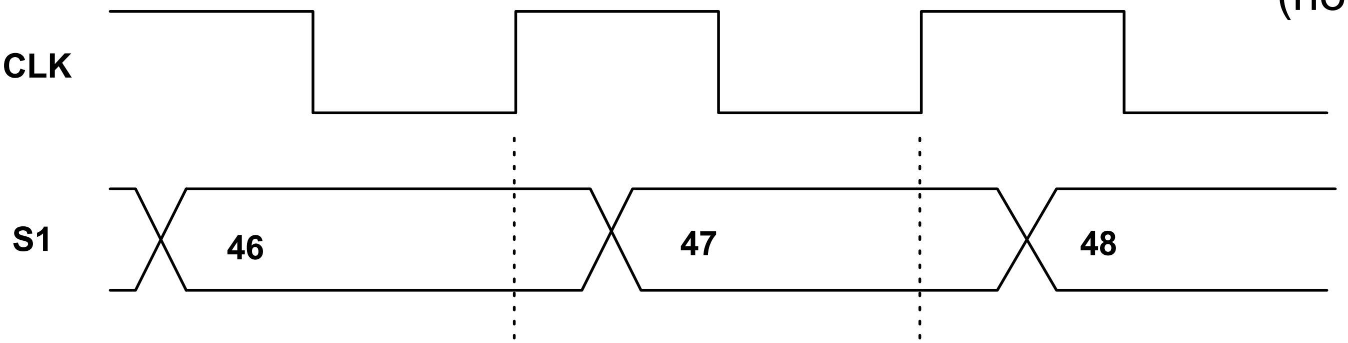
suppose B arrives *just* late enough to avoid a hold time violation

what if CLKB is *delayed* with respect to CLKA?

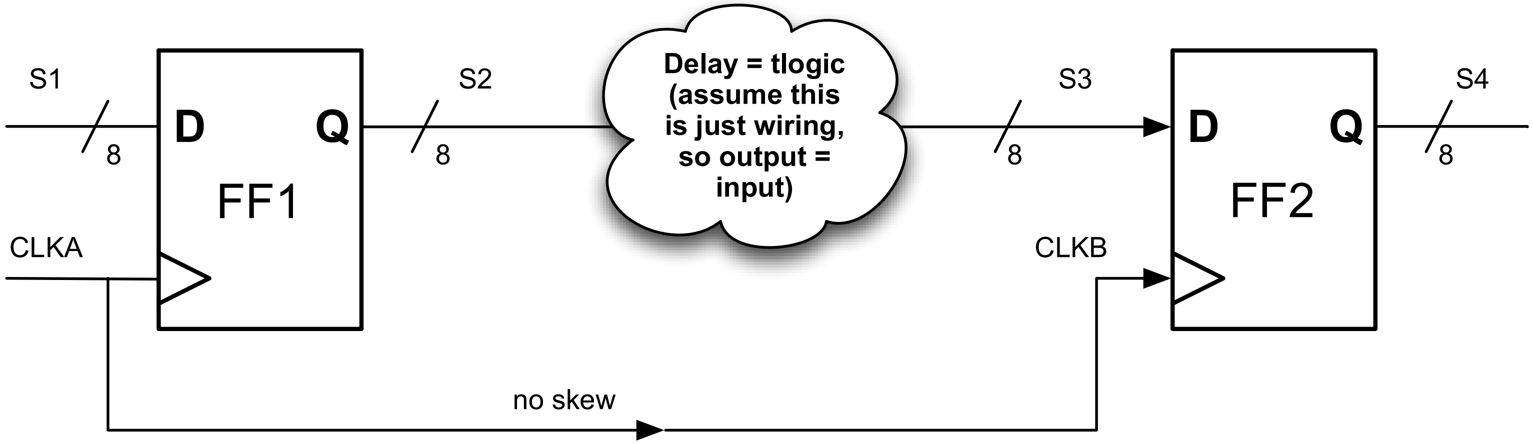
Hold time example:


S1
8
D
Q
FF1
CLKA
S2
8
Delay = tlogic (assume this is just wiring, so output = input)
S3
8
D
Q
FF2
CLKB
S4
8
no skew

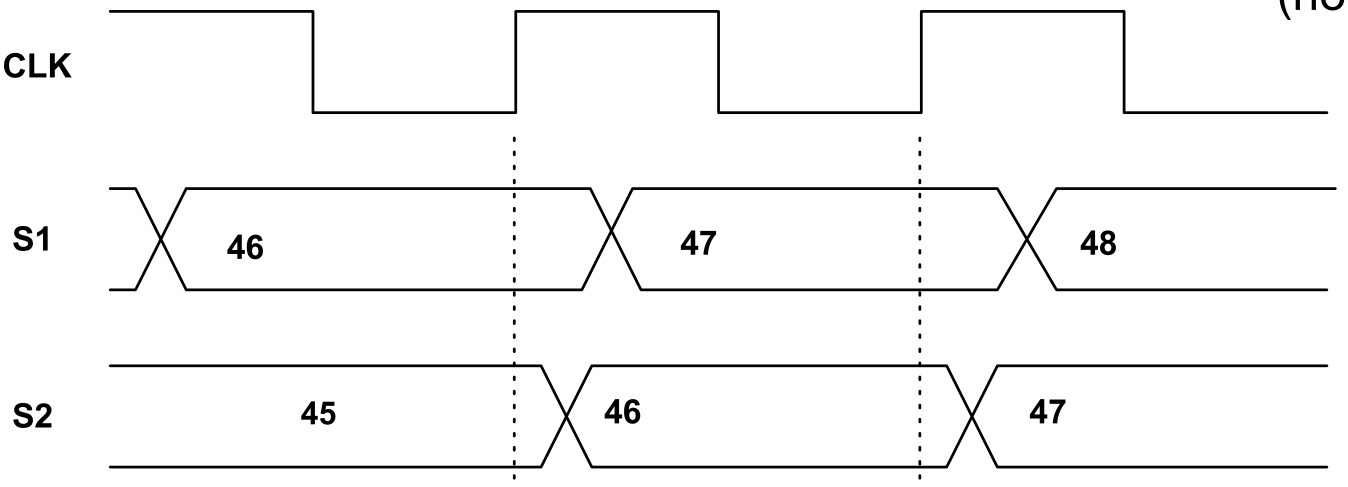
Correct Operation
(no skew)



Hold time example:

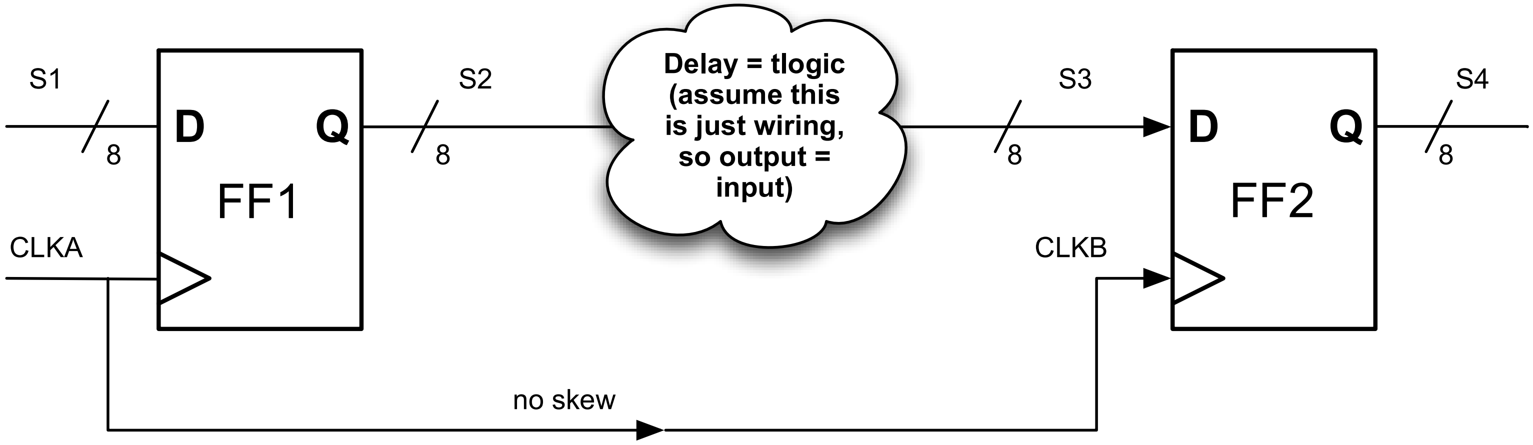

S1
8
D
Q
FF1
CLKA
S2
8
Delay = tlogic (assume this is just wiring, so output = input)
S3
8
D
Q
FF2
CLKB
S4
8
no skew

Correct Operation
(no skew)

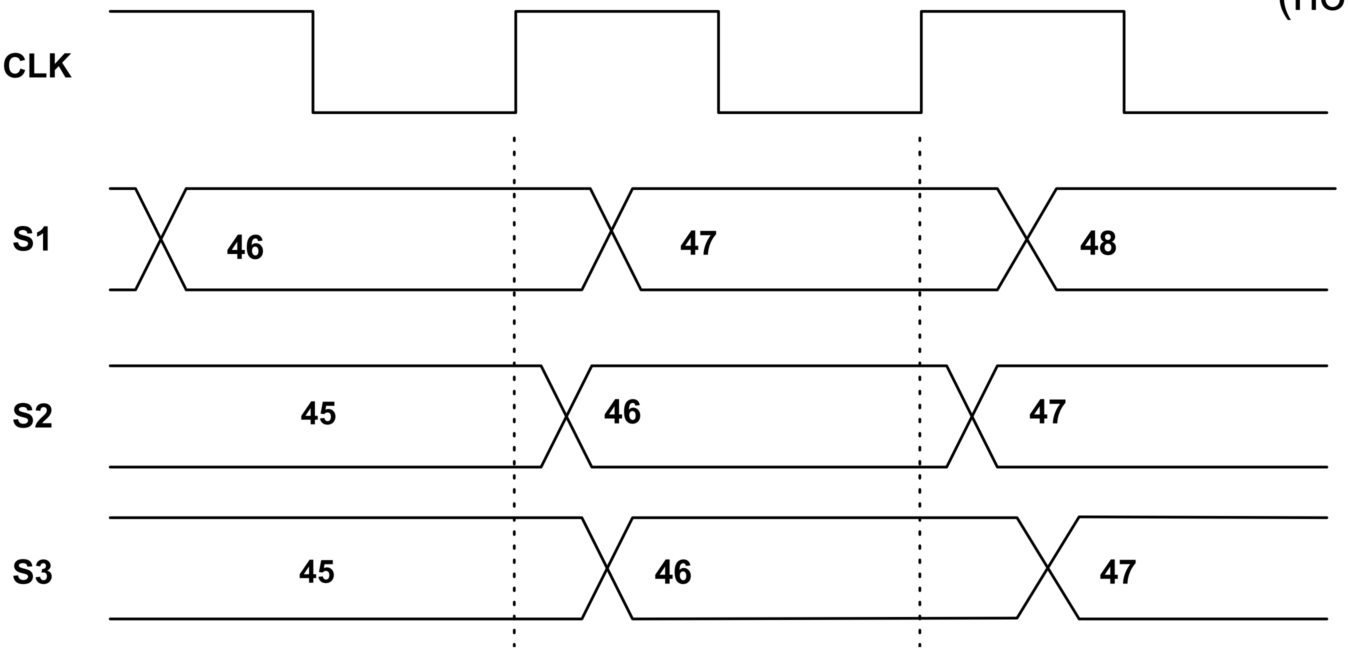


FF1 Latches 46
@ edge 1

Hold time example:


S1
8
D
Q
FF1
CLKA
S2
8
Delay = tlogic (assume this is just wiring, so output = input)
S3
8
CLKB
D
Q
FF2
S4
8
no skew

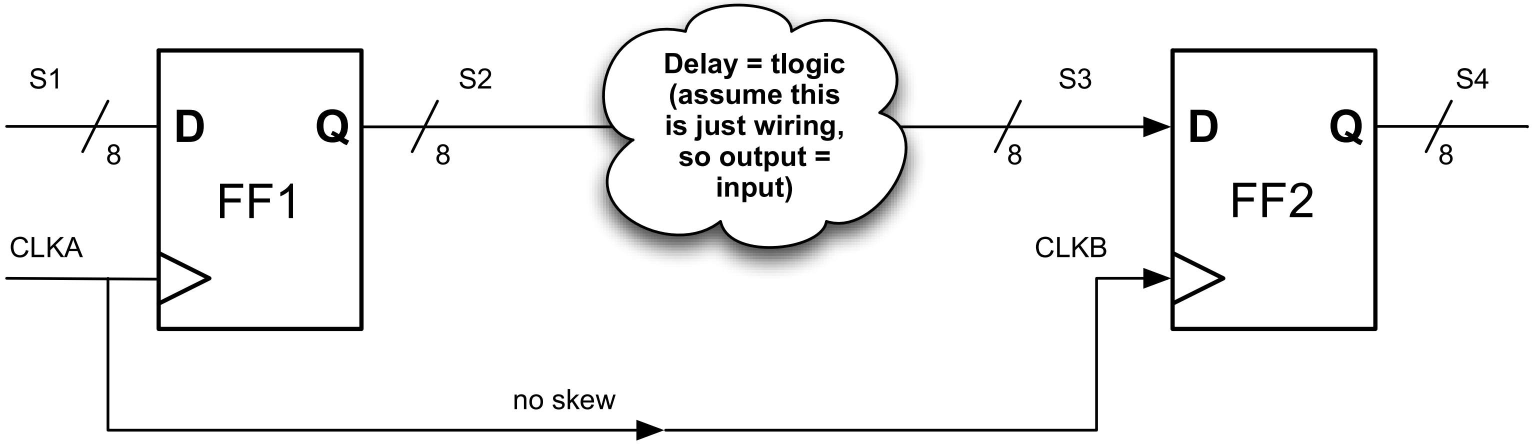
Correct Operation
(no skew)



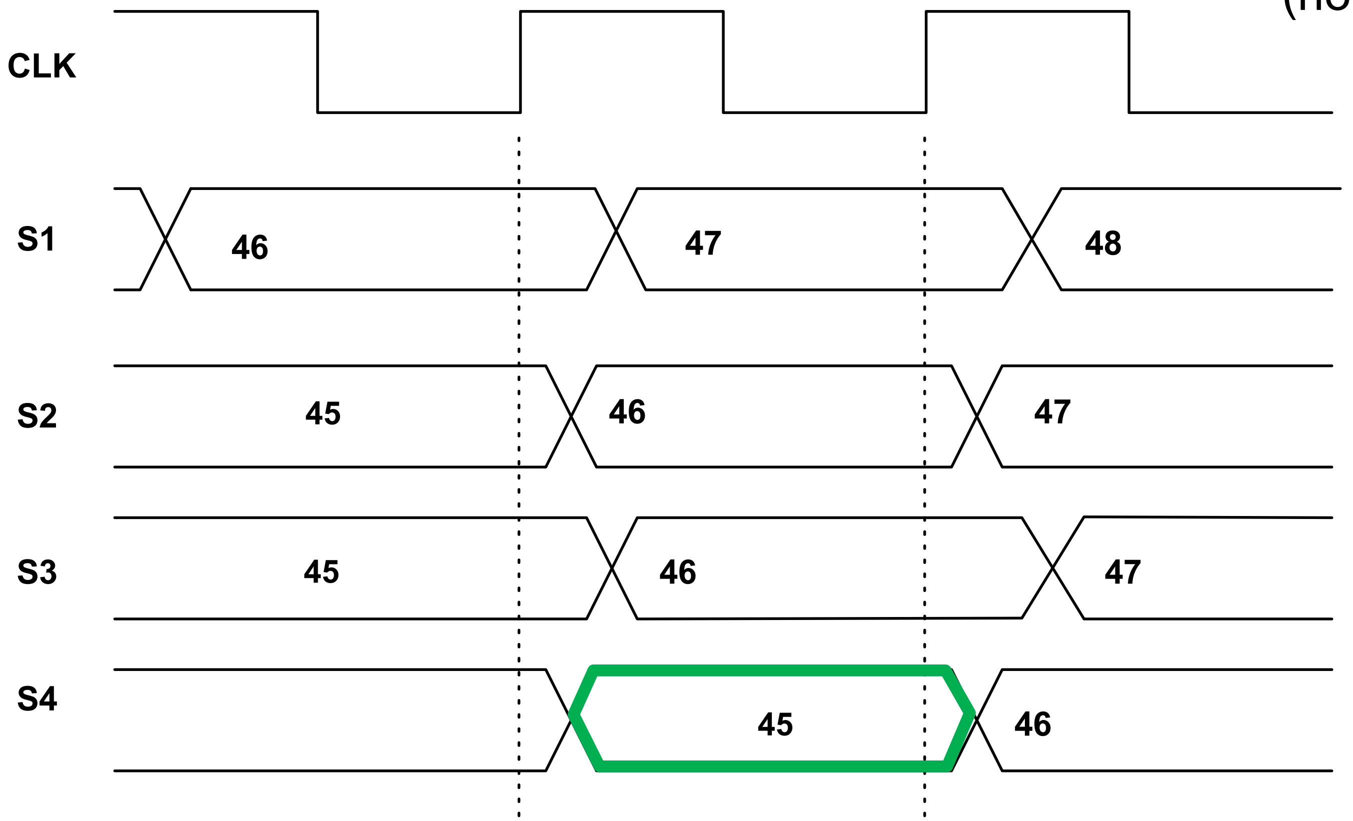
FF1 Latches 46
@ edge 1

FF2 Inputs 45
@ edge 1

Hold time example:


S1
8
D
Q
FF1
CLKA
S2
8
Delay = tlogic (assume this is just wiring, so output = input)
S3
8
D
Q
FF2
CLKB
S4
8
no skew

Correct Operation
(no skew)

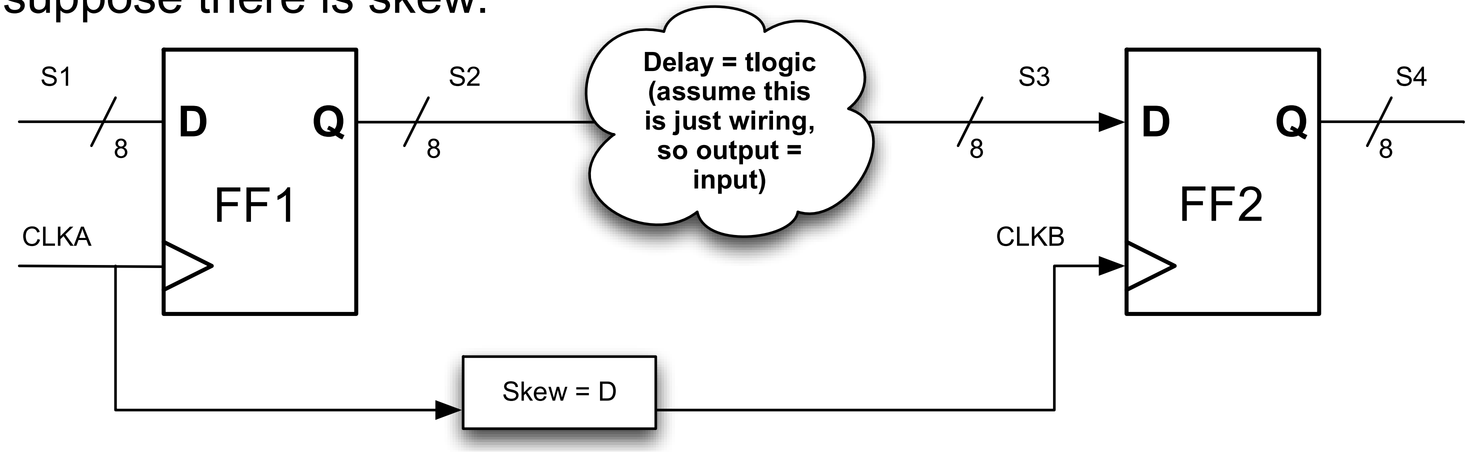


FF1 Latches 46
@ edge 1

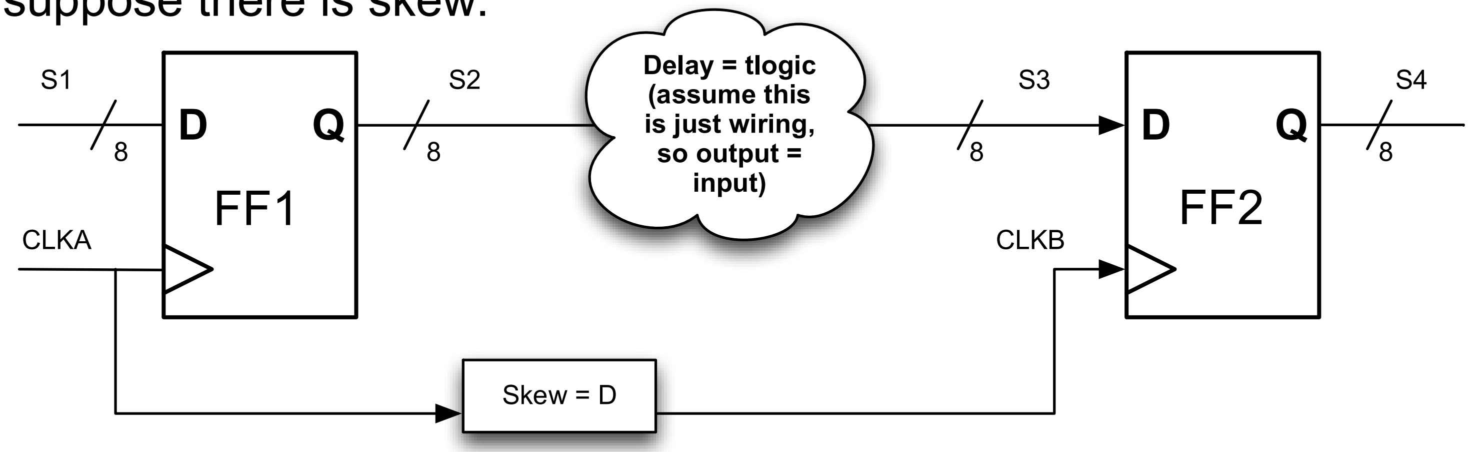
FF2 Inputs 45
@ edge 1

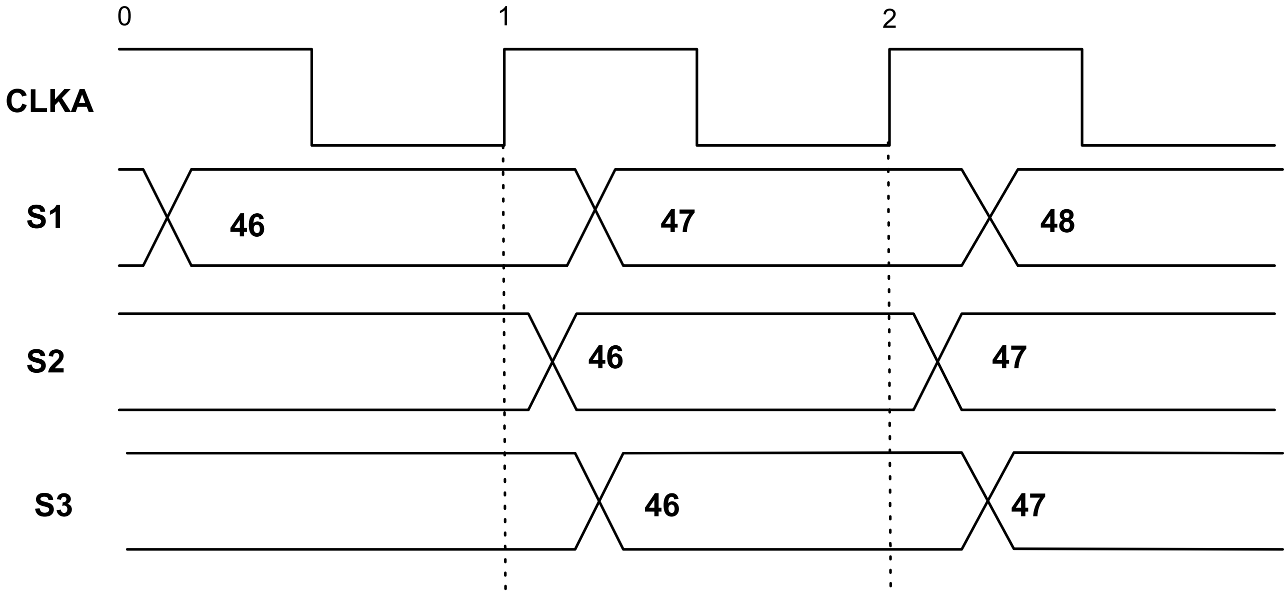
FF2 Latches 45
@ edge 1

Now suppose there is skew:



Now suppose there is skew:

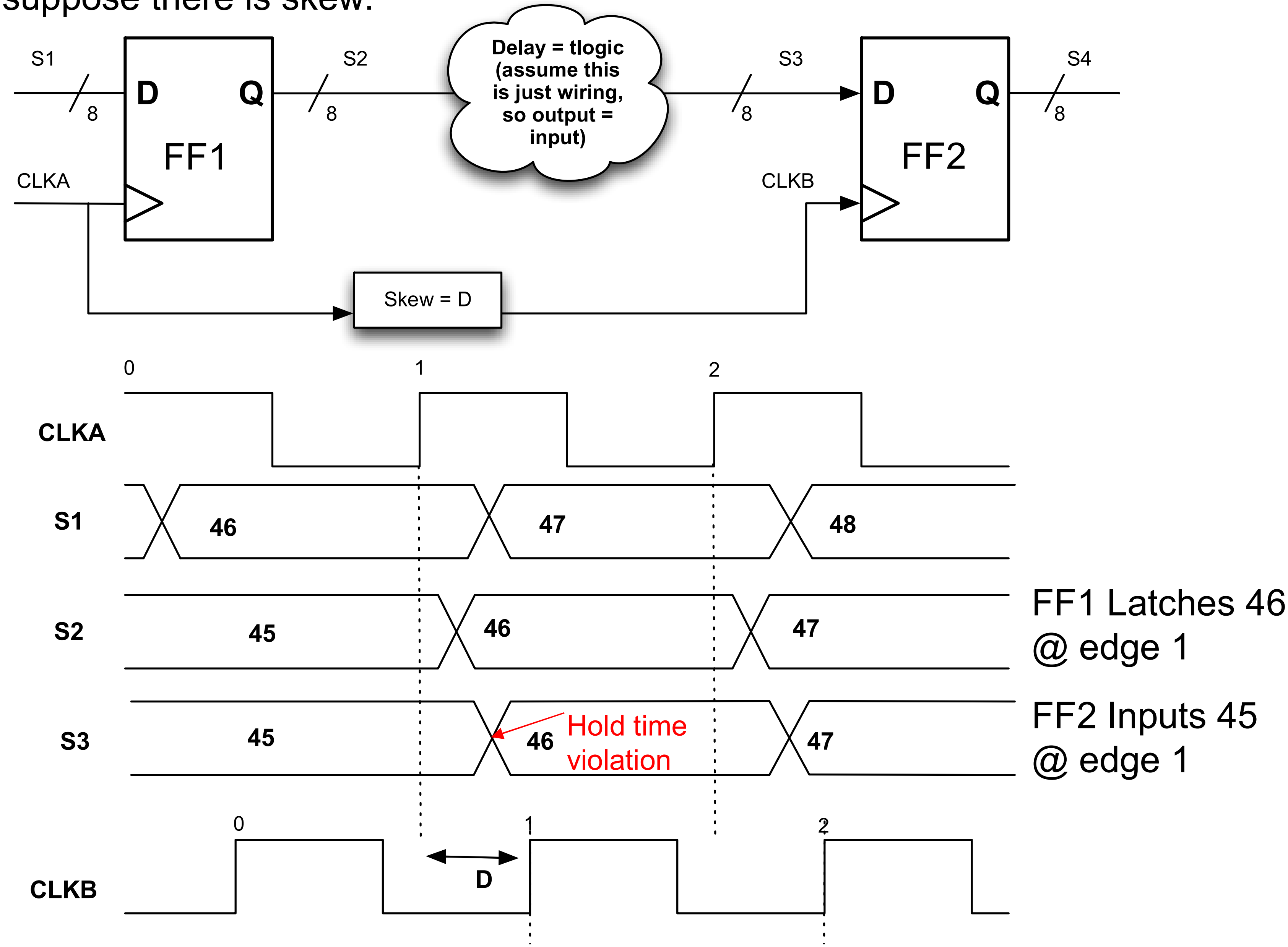

S1
8
D
Q
FF1
CLKA
S2
8
Delay = tlogic (assume this is just wiring, so output = input)
S3
8
D
Q
FF2
CLKB
S4
8
Skew = D



FF1 Latches 46 @ edge 1

FF2 Inputs 45 @ edge 1

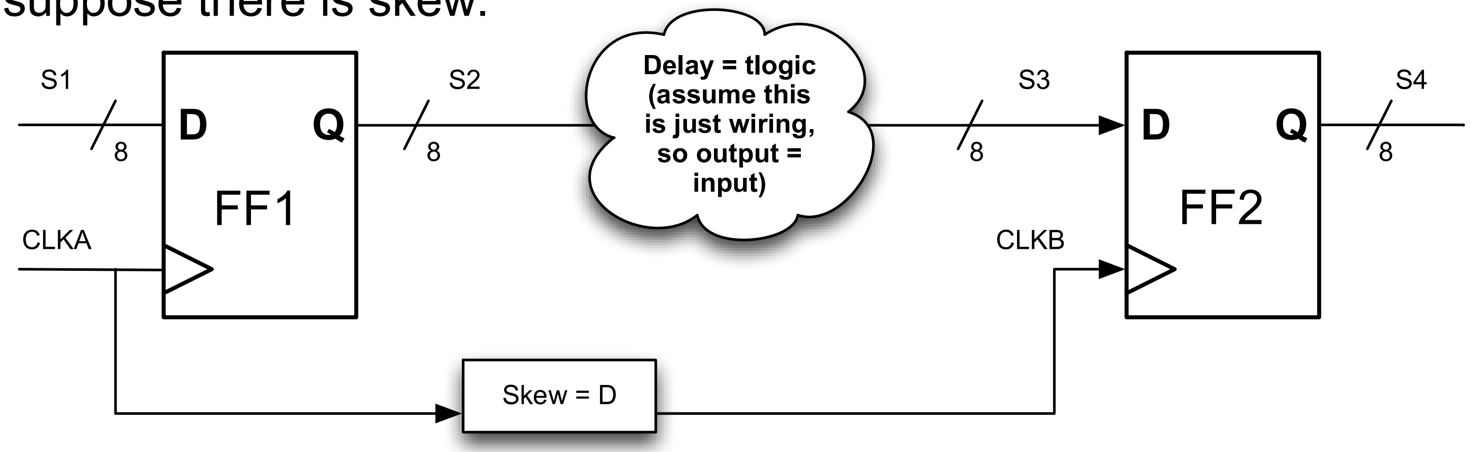
Now suppose there is skew:

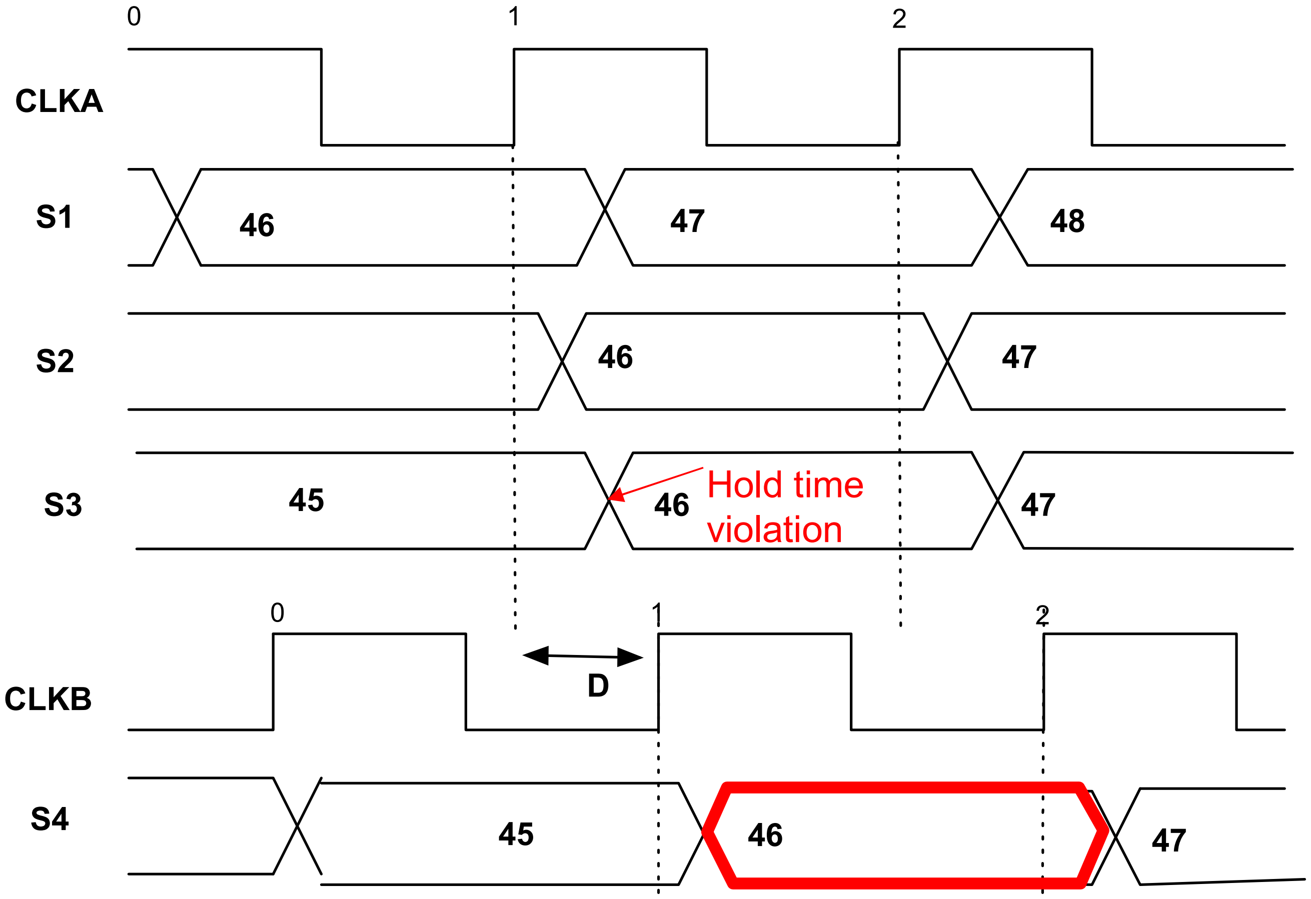


FF1 Latches 46 @ edge 1

FF2 Inputs 45 @ edge 1

Now suppose there is skew:

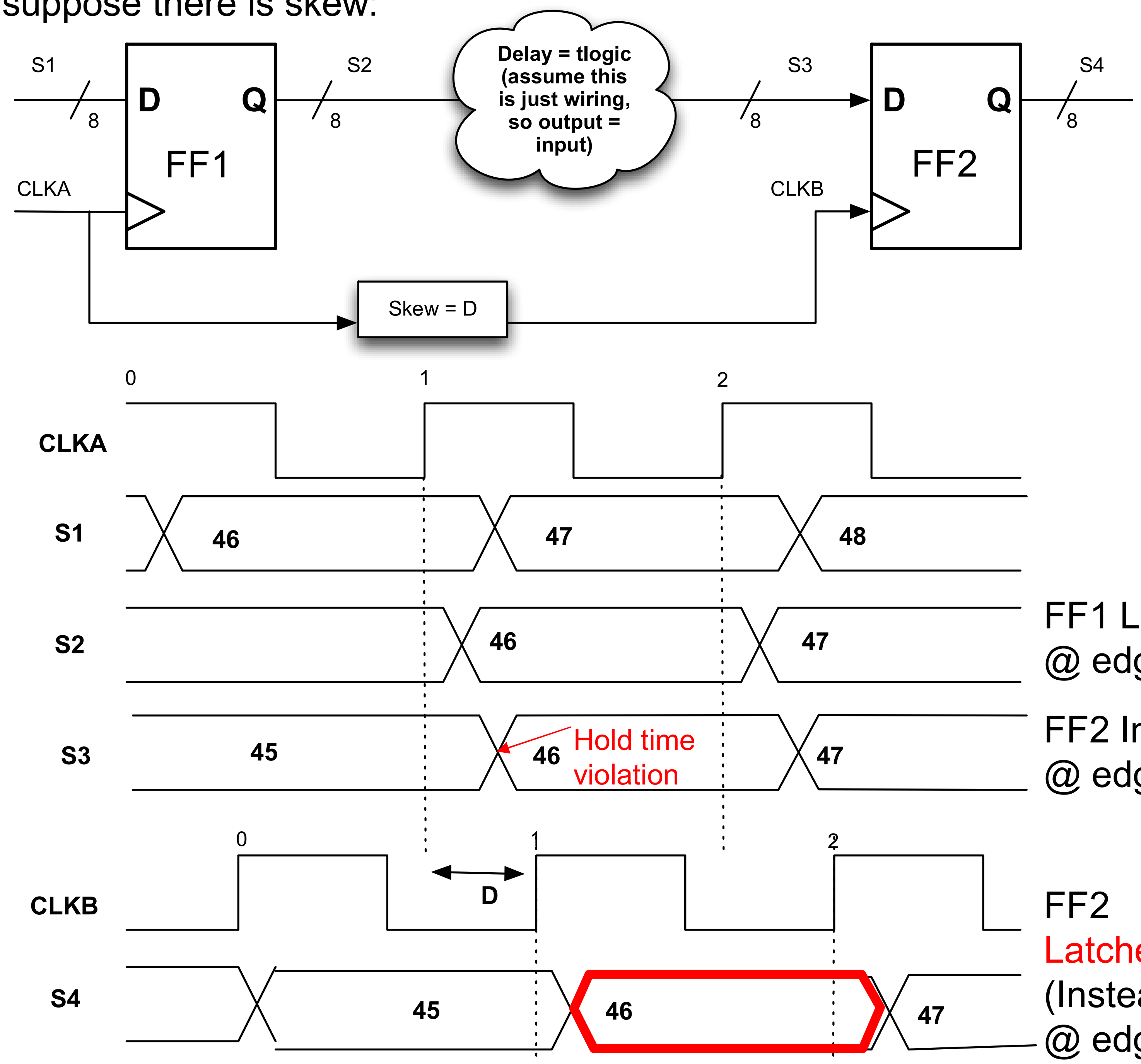

S1
8
D
Q
FF1
CLKA
S2
8
Delay = tlogic (assume this is just wiring, so output = input)
S3
8
D
Q
FF2
CLKB
S4
8
Skew = D



FF1 Latches 46 @ edge 1

FF2 Inputs 45 @ edge 1

Now suppose there is skew:



FF1 Latches 46
@ edge 1

FF2 Inputs 45
@ edge 1

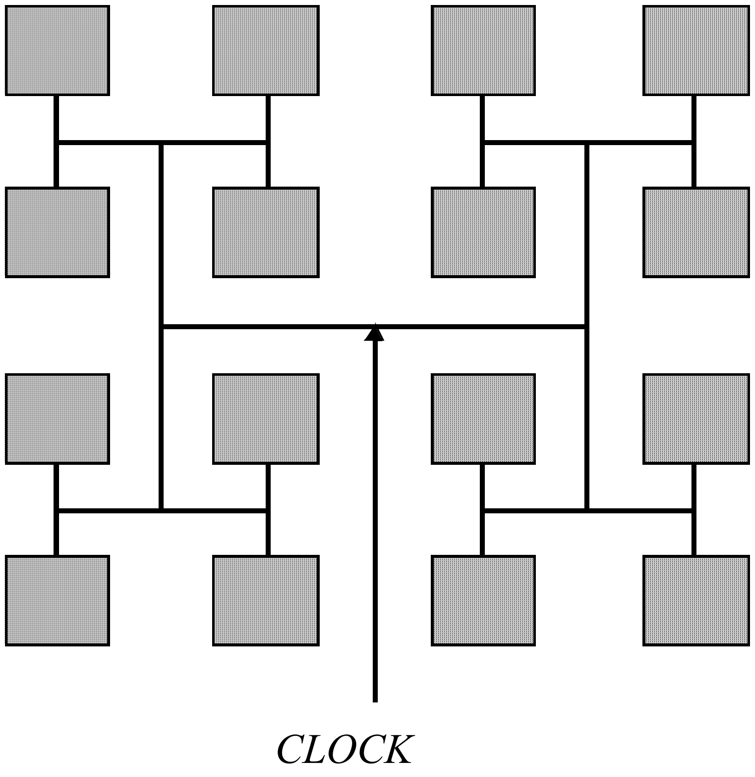
FF2
Latches 46
(Instead of 45)
@ edge 1+D

In this example, skew caused a hold violation, causing the second value “46” to race through too early. This caused a functional violation.

Slowing the clock speed doesn’t help.

Lesson: Clock skew needs to be avoided as much as possible

Clock Skew: What can we do? Custom Chip



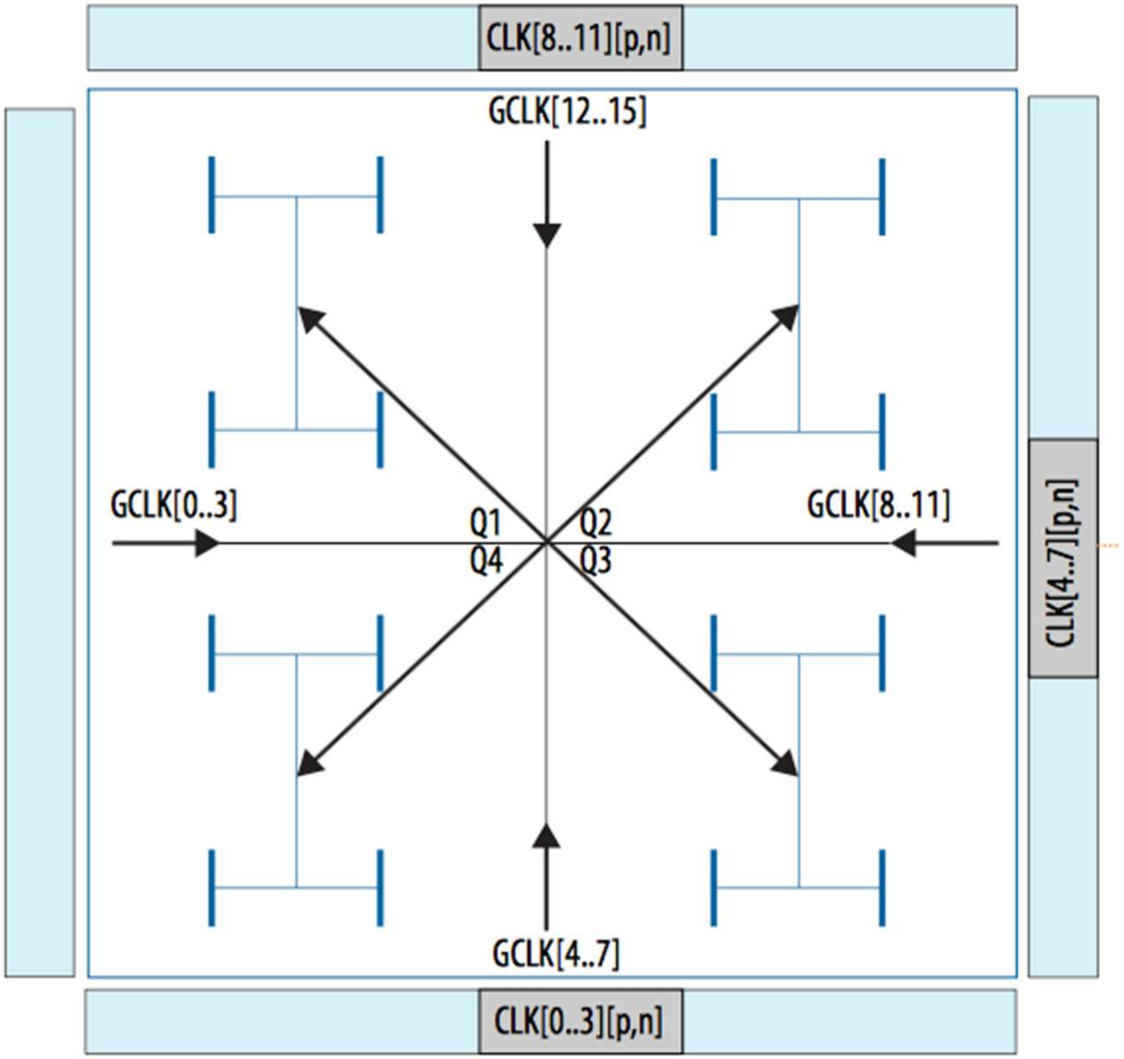
H-Tree Network

Observe: Only Relative Skew is Important

Layout clock signal to minimize skew

Need to verify timing extensively to make sure skew is not going to cause failure.

Clock Skew: FPGAs



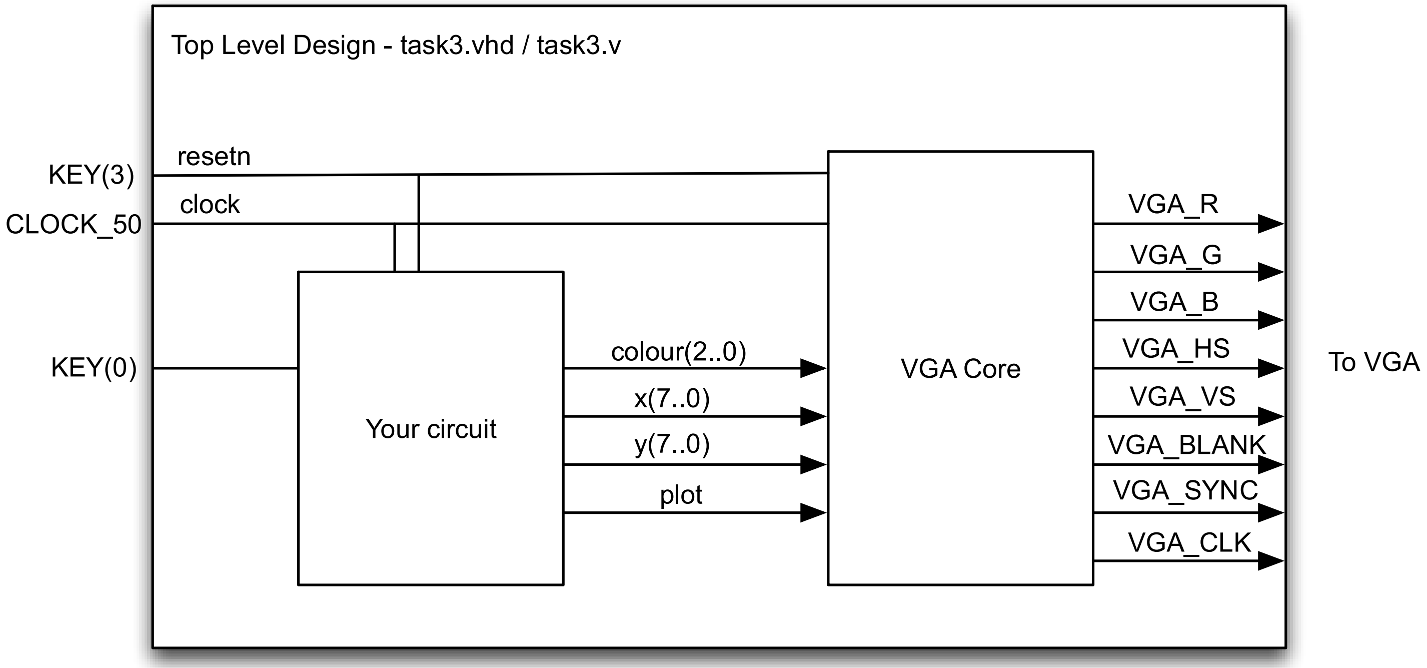
FPGAs have complex clock distribution networks to achieve low skew

CAD tools must verify against clock skew problems.

From Cyclone V handbook

PLLs: Motivation

Consider the VGA core from earlier lab:



Your chip receives a 50 MHz clock

The VGA spec says VGA_CLK needs to run at 25 MHz*

The designer of the VGA core had to make a 25MHz clock out of a 50MHz clock

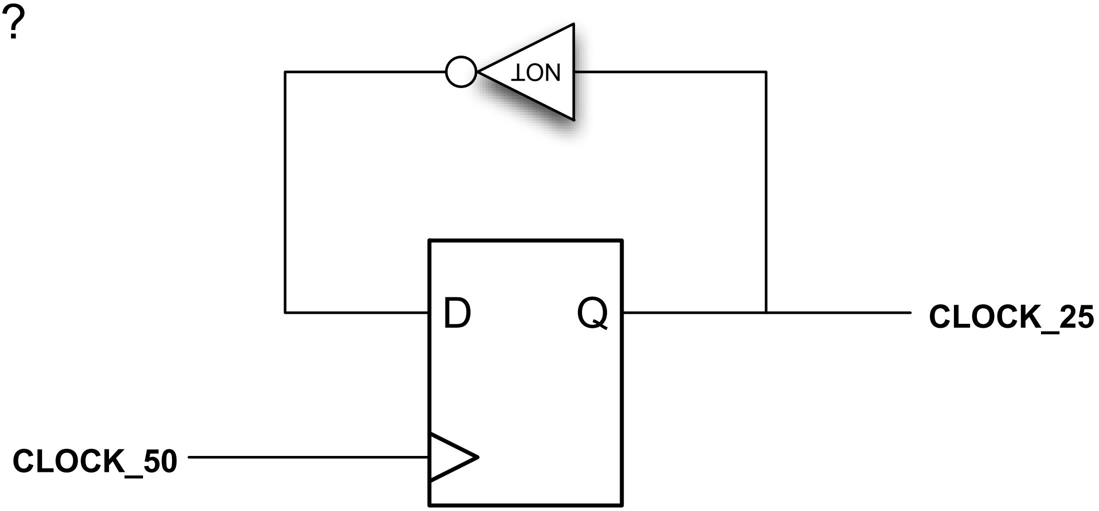
*actually 25.175 MHz

PLLs: Motivation

How would you make a circuit that creates a 25 MHz clock from a 50 MHz clock?

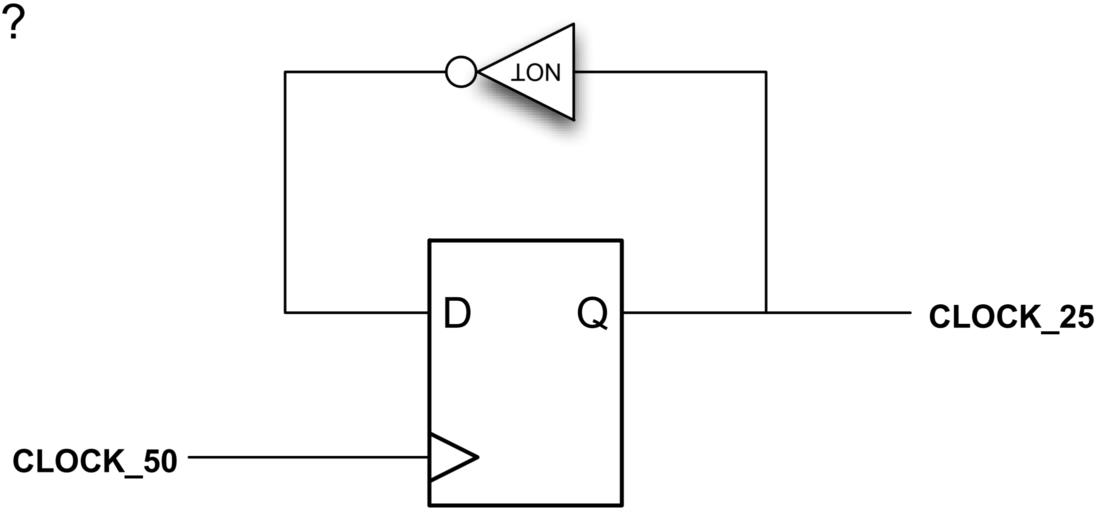
PLLs: Motivation

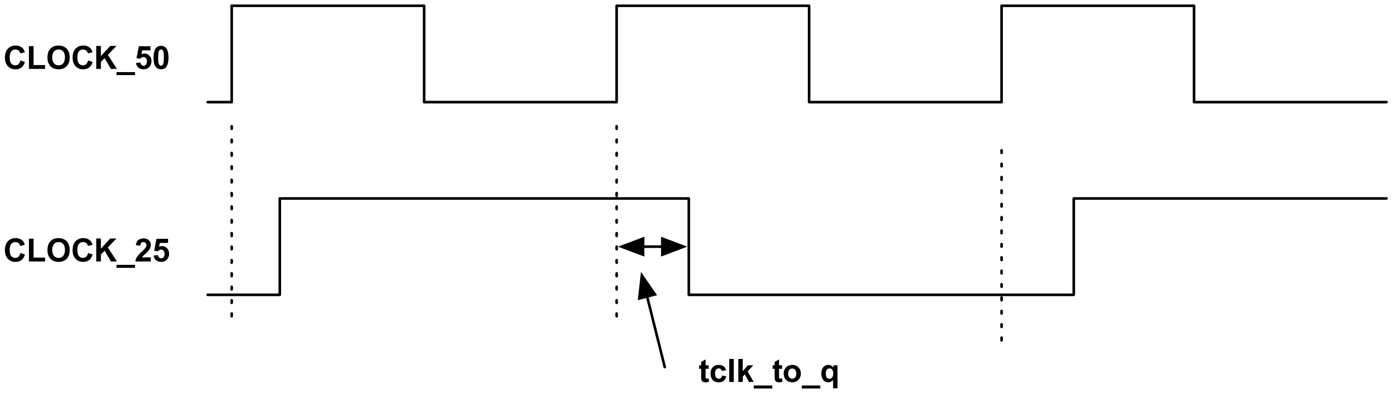
How would you make a circuit that creates a 25 MHz clock from a 50 MHz clock?



PLLs: Motivation

How would you make a circuit that creates a 25 MHz clock from a 50 MHz clock?



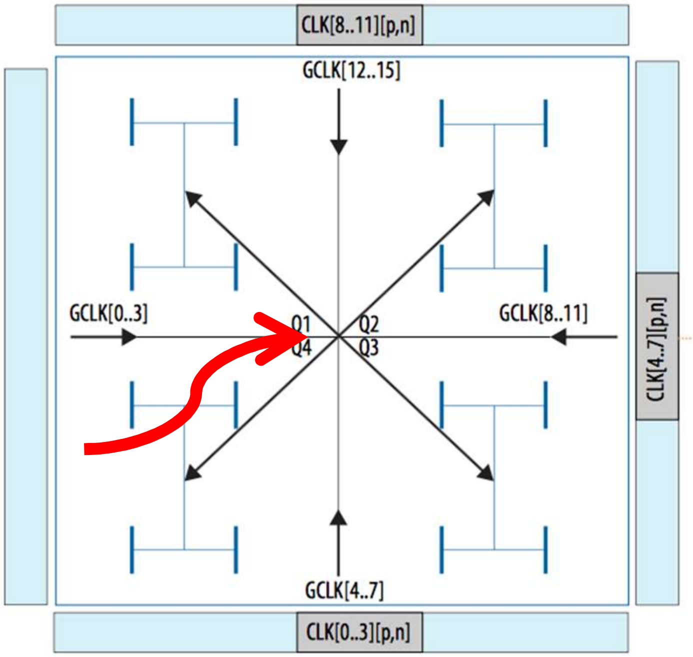


PLLs: Motivation

Two problems with this solution:

1. Clock skew between CLOCK_50 and CLOCK_25

2. Even if you use only CLOCK_25, it must be distributed across the entire chip. FPGA routing is slow and unpredictable.

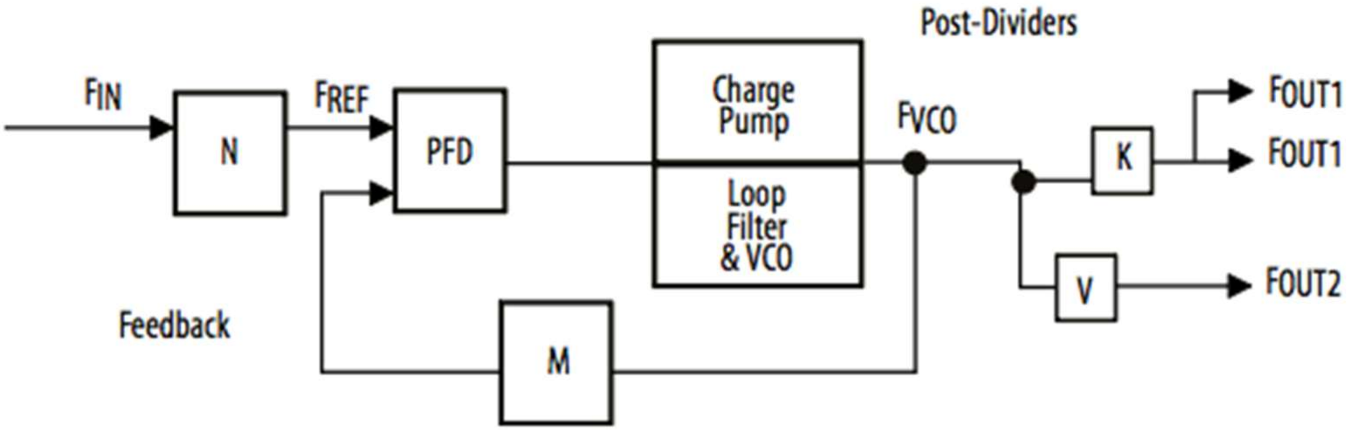


Even if CLOCK_25 uses the global clock distribution network, we still have to route from our FF output to the source of these wires.

Phase Locked Loops

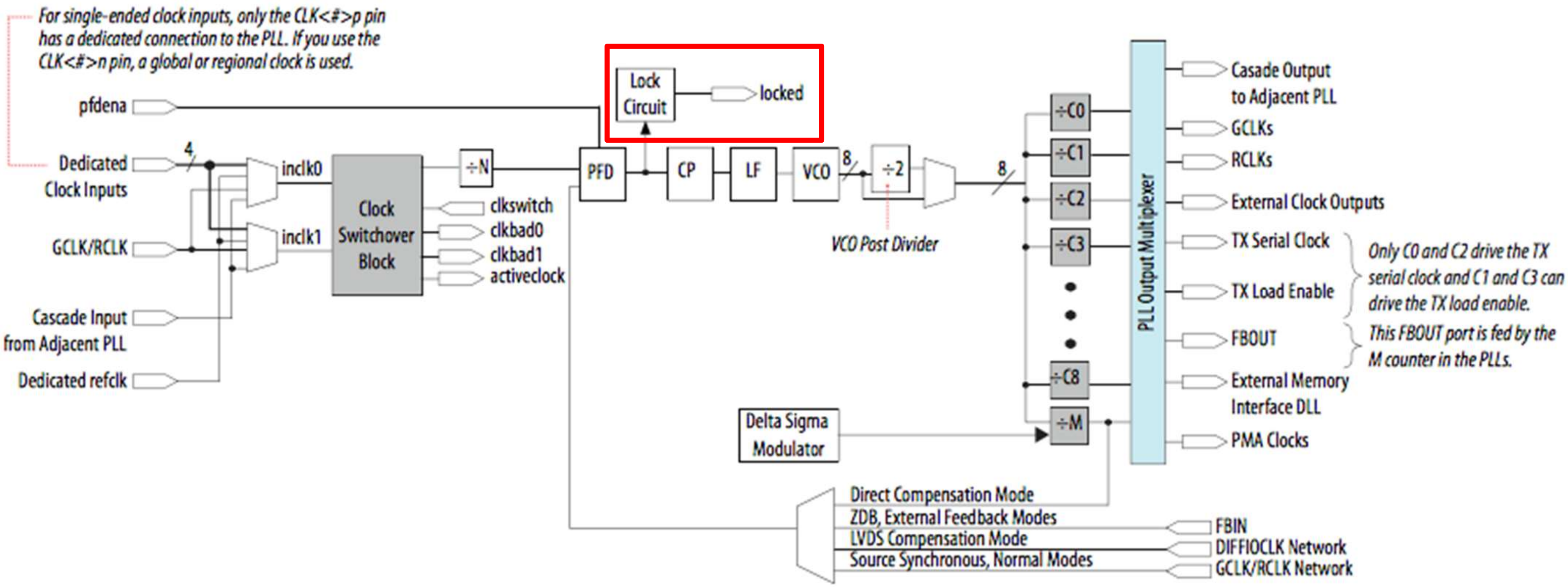
Modern FPGAs contain Phase Locked Loop hard cores:

- A mixed-signal circuit (= both analog and digital) that generates output clocks aligned to an input clock
- Can act as a clock speed divider or multiplier
- Can directly feed clock distribution network
- Automatically adjusts phase to account for clock distribution network



From ALTPLL IP Core User Guide

Cyclone V PLL: Detail


For single-ended clock inputs, only the CLK<#>p pin has a dedicated connection to the PLL. If you use the CLK<#>n pin, a global or regional clock is used.
pfdena
Dedicated Clock Inputs
4
GCLK/RCLK
Cascade Input from Adjacent PLL
Dedicated refclk
inclk0
inclk1
Clock Switchover Block
÷N
clkswitch
clkbad0
clkbad1
activeclock
Lock Circuit
locked
PFD
CP
LF
VCO
8
÷2
VCO Post Divider
8
÷C0
÷C1
÷C2
÷C3
÷C8
÷M
Delta Sigma Modulator
PLL Output Multiplexer
Casade Output to Adjacent PLL
GCLKs
RCLKs
External Clock Outputs
TX Serial Clock
TX Load Enable
FBOUT
External Memory Interface DLL
PMA Clocks
Only C0 and C2 drive the TX serial clock and C1 and C3 can drive the TX load enable.
This FBOUT port is fed by the M counter in the PLLs.
Direct Compensation Mode
ZDB, External Feedback Modes
LVDS Compensation Mode
Source Synchronous, Normal Modes
FBIN
DIFFIOCLK Network
GCLK/RCLK Network

From Cyclone V Handbook

Phase Locked Loops

If you look inside vga_pll.v and vga_adapter.v :

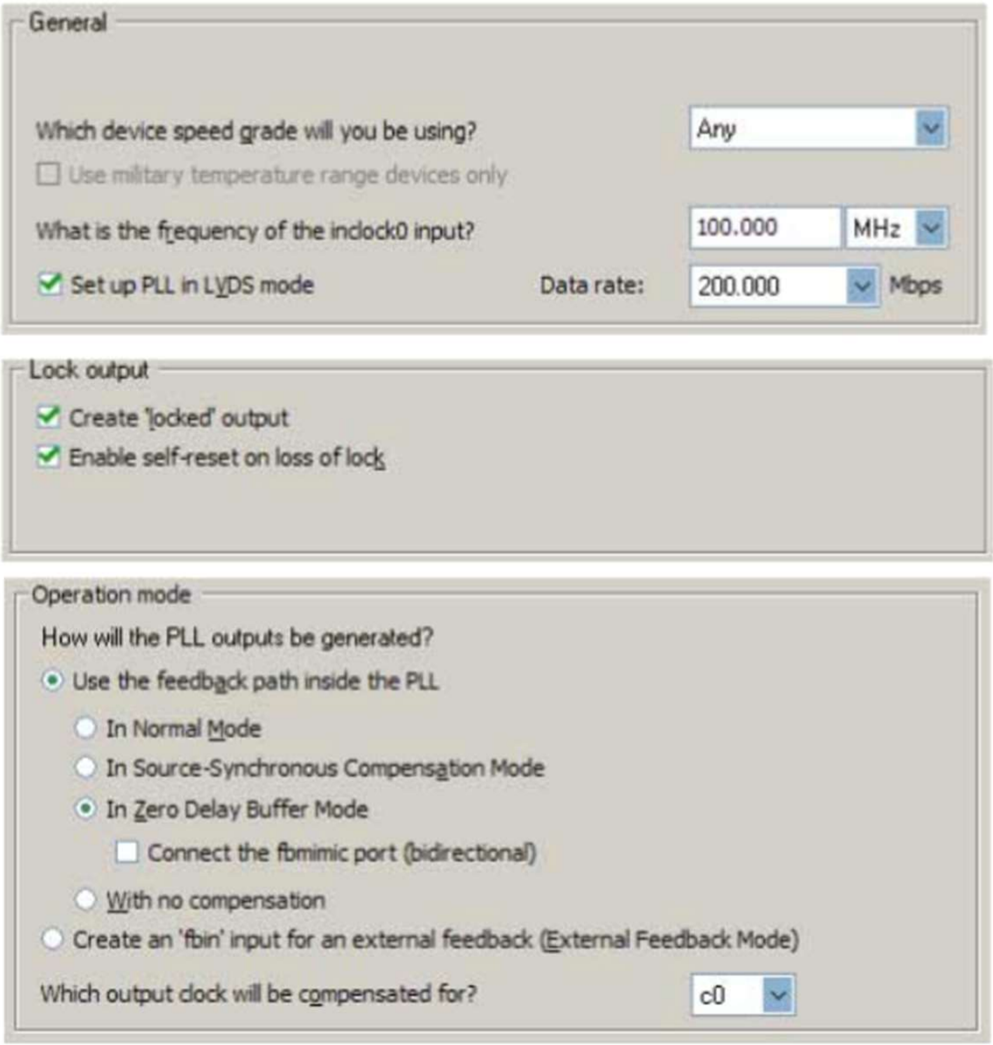
```
altpll  altpll_component (.inclk (clock_input_bus),
         .clk (clock_output_bus));
    defparam
        altpll_component.operation_mode = "NORMAL”,
        altpll_component.intended_device_family = "Cyclone
II",

        altpll_component.lpm_type = "altpll”,
        altpll_component.pll_type = "FAST”,
        altpll_component.inclk0_input_frequency = 20000,
        altpll_component.primary_clock = "INCLK0",
        altpll_component.compensate_clock = "CLK0”,
        altpll_component.clk0_phase_shift = "0”,
        altpll_component.clk0_divide_by = 2,
        altpll_component.clk0_multiply_by = 1,
```

```
vga_pll mypll (CLOCK_50, clock_25);
```

Used to divide the clock frequency by 2. Can also multiply clock frequency.
Don’t worry about details. If you ever need it, you can look it up

Many PLL Parameters



General

Which device speed grade will you be using? Any

□ Use military temperature range devices only

What is the frequency of the inclock0 input? 100.000 MHz

☑ Set up PLL in LVDS mode Data rate: 200.000 Mbps

Lock output

☑ Create 'locked' output

☑ Enable self-reset on loss of lock

Operation mode

How will the PLL outputs be generated?

● Use the feedback path inside the PLL

○ In Normal Mode

○ In Source-Synchronous Compensation Mode

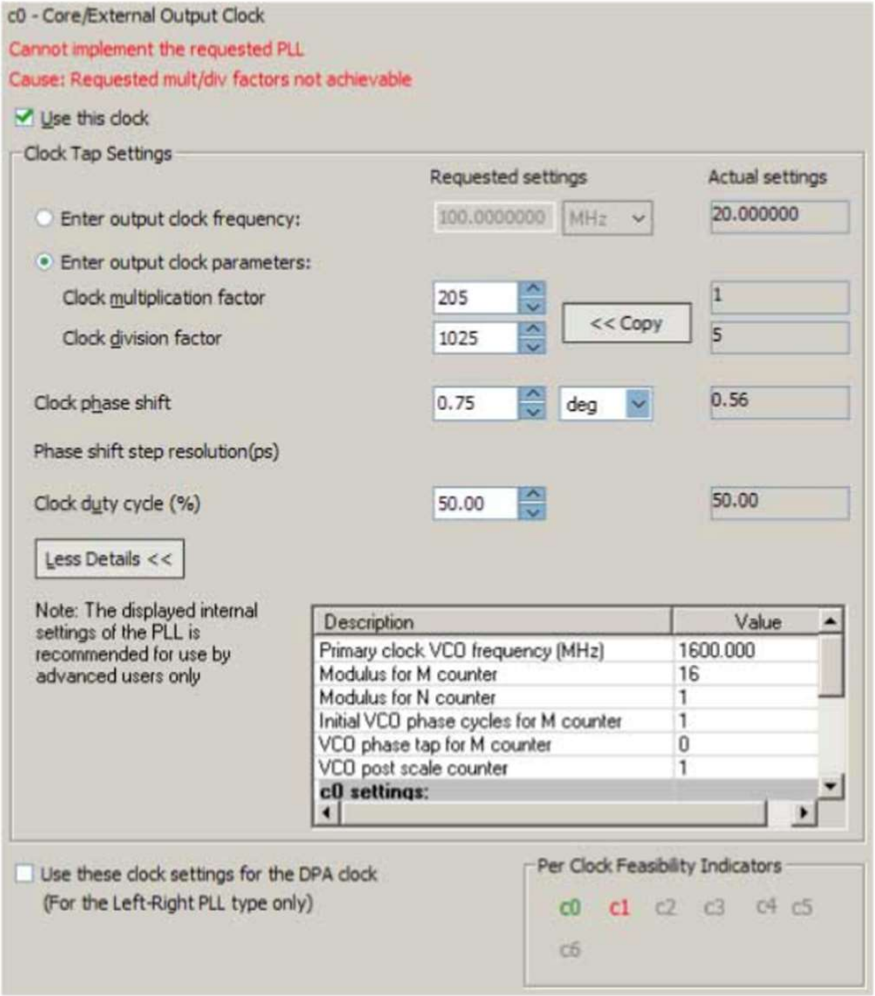
● In Zero Delay Buffer Mode

□ Connect the fbmimic port (bidirectional)

○ With no compensation

○ Create an 'fbin' input for an external feedback (External Feedback Mode)

Which output clock will be compensated for? c0



c0 - Core/External Output Clock

Cannot implement the requested PLL

Cause: Requested mult/div factors not achievable

☑ Use this clock

Clock Tap Settings

	Requested settings		Actual settings
○ Enter output clock frequency:	100.0000000	MHz	20.000000
● Enter output clock parameters:			
Clock multiplication factor	205	<< Copy	1
Clock division factor	1025		5
Clock phase shift	0.75	deg	0.56
Phase shift step resolution(ps)			
Clock duty cycle (%)	50.00		50.00

Note: The displayed internal settings of the PLL is recommended for use by advanced users only

Description	Value
Primary clock VCO frequency (MHz)	1600.000
Modulus for M counter	16
Modulus for N counter	1
Initial VCO phase cycles for M counter	1
VCO phase tap for M counter	0
VCO post scale counter	1
c0 settings:	

□ Use these clock settings for the DPA clock
(For the Left-Right PLL type only)

Per Clock Feasibility Indicators

(parameter values shown are just examples, not recommendations)

Glitches

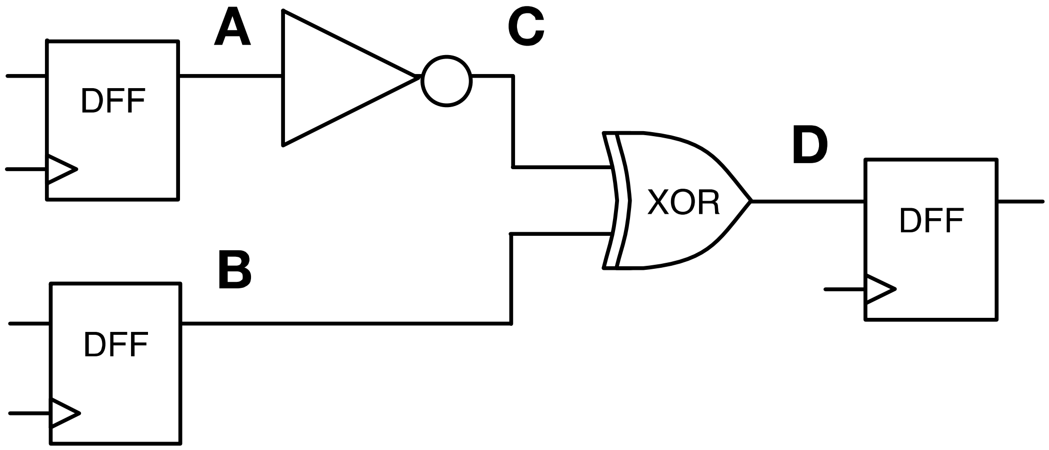
Glitch

An undesired short-lived pulse that occurs before a signal settles to its intended value

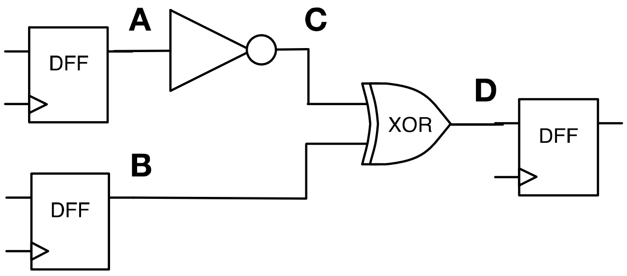
Causes

- Unequal arrival times of inputs on combinational gates
- Various electrical effects (eg crosstalk, not covered in this course)

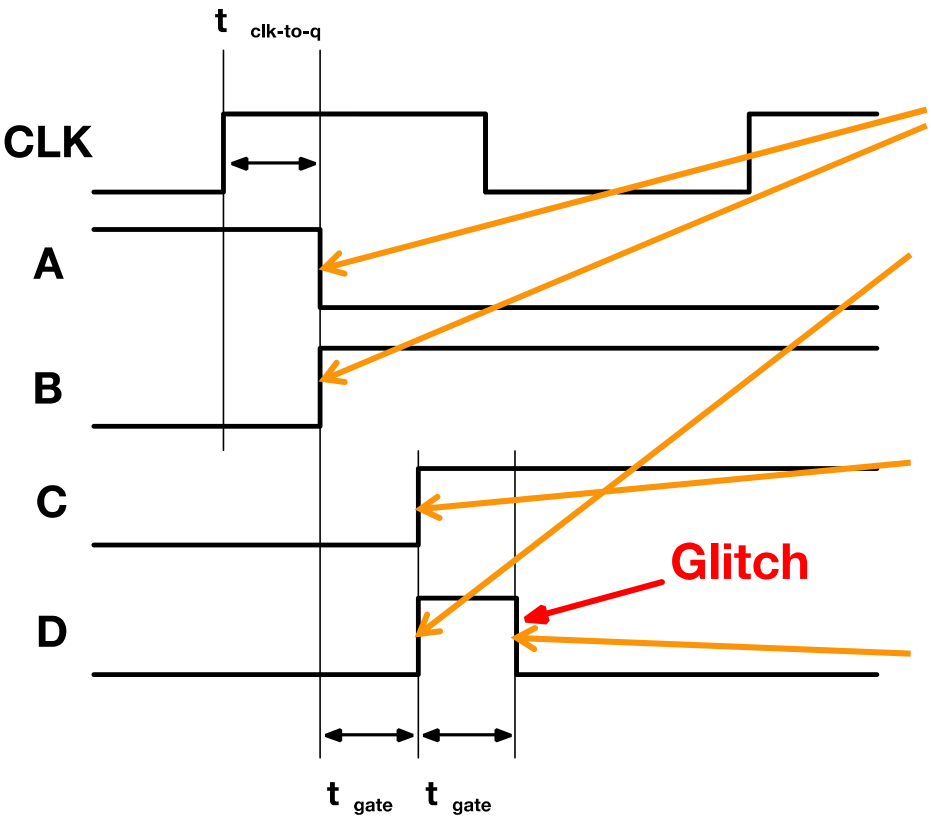
Example of a circuit with glitching:



Glitch Example



B	C	D
0	0	0
0	1	1
1	0	1
1	1	0



1) A, B settle after clock-to-q delay

2a) B arrives immediately at XOR, causing a transition to 1 on D after one gate delay

2b) At the same time, A arrives at the INV immediately, causing a transition to 1 on C after one gate delay

3) After C settles, its value arrives at the XOR after one additional gate delay, causing a transition back to 0 on D

Glitches – Key Take Aways

A signal may switch several times before settling to final intended value

- Source (new glitch): uneven signal arrival times on inputs
- Propagate (existing glitch): input glitch → output glitch

Glitches are normal …

ok if signal is stable before anyone tries to do something with data

eg, before t_{setup} of flip-flop

Power and Energy

- Glitches cause extra charge/discharge cycles of output capacitance
- Unnecessary power consumption

Learning Objectives

1. Understand what timing closure is and why it is difficult
2. Understand how pipelining can help with timing closure
3. Understand retiming and be able to apply it to a circuit
4. Be able to discuss the effects of clock skew
5. Understand what a PLL is used for
6. Understand the cause and impact of glitches caused by unequal combinational path delays